Module 4: *Disease*

Team Members:

Addison and Yaseen

Project Title:

SIR Analysis of the 2009 Swine Flu Epidemic in Mexico

Project Goal:

This project seeks to predict cases of swine flu in Mexico 2009 based on a cumultative data set using SIR Analysis, Euler's, and RK methods.

Disease Background:

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

Prevalence & incidence

- The incidence (or new cases per day) of Swine Flu in Mexico 2009 started at a few cases per day in late April, hundereds of new cases per day in May and June, and thousands of new cases per day by early July. The prevalence, or number of infected people at any given time was smaller compared to other diseases. This is because the infection period for Swine Flu is about 5-7 days (a value that is used in the code below). At it's peak, the prevalence of Swine Flu was in the thousands (Chat GPT prompt: What was the prevalence and incidence of Swine Flu in Mexico 2009?). By the end of 2009, there were a total 27,440 confirmed cases of Swine Flu, with 585 deaths (https://pmc.ncbi.nlm.nih.gov/articles/PMC4957980/) *Economic burden*
- The financial impact of Swine Flu as of May 2009 was 2 billion dollars to the Mexican economy. At the time, it was predicted that the GDP would decrease by 4.8 percent, however businesses in Mexico were able to reopen relatively early around mid-May (https://www.npr.org/sections/health-shots/2009/05/swine_flu_hits_mexico_economy.html). The Swine Flu pandemic significantly impacted port trade, causing Mexico to have a pork trade deficit of 27 million dollars at the end of 2009. Additionally, the tourism industry was significantly impacted, with the loss of visitors causing losses of around 2.8 billion dollars by the end of 2009 (https://pubmed.ncbi.nlm.nih.gov/23744805/). *Risk factors (genetic, lifestyle) & Societal determinants*
- One risk factor for Swine Flu is age. Outcomes are generally worse in children under the age of 2 and adults over the age of 65. Living or working conditions may also impact someones susceptibility to H1N1. Those who live with a lot of people, such as those in nursing homes, military housing, or hospitals are at a higher risk. Also, those with weaker immune systoms are at a higher risk. This could be due to cancer treatments, long-term steroid use, organ transplants, or HIV/AIDS. Chronic illnesses have similar effects. American Indians or Alaskan Natives may have a higher risk of complications, in addition to those who are pregnant or those with a BMI over 40. Finally, people who are under the

age of 19 and are under aspirin therapy may have a higher risk of complications due to H1N1. *Symptoms*

- Symptoms of Swine Flu are similar to symptoms of other influenza viruses. Symptoms include fever, muscle aches, chills, sweats, cough, sore throat, runny/stuffy nose, watery or red eyes, eye pain, tiredness, diarrhea, and vomiting. Symptoms start 1-4 days after exposure. It is recommended to see a doctor if you also have a pre-existing condition that may impact your health or experience severe symptoms. (https://www.mayoclinic.org/diseases-conditions/swine-flu/symptoms-causes/syc-20378103) *Diagnosis*
- Health care providers may do a physical assessment in order to diagnose a patient, or may perform a test. Tests are more likely to occur if you are already at the hospital, are at higher risk for complications, or live with someone who is at a higher risk of complications. The test in order to diagnose Swine Flu is a PCR test. (https://www.mayoclinic.org/diseases-conditions/swine-flu/diagnosis-treatment/drc-20378106) *Biological mechanisms (anatomy, organ physiology, cell & molecular physiology)*
- H1N1 typically infects humans through respiratory droplets emitted when one coughs or sneezes, and enters through the nasal cavity, through, and lungs. It targets the upper and lower respiratory tract, including the nasal mucosa, pharynx, trachea, bronchi, bronchioles, and alveoli. When the virus interacts with cells, the Hemagglutinin of the virus binds to sialic acid residues on respiratory epithelial cells and is taken up into the cell via endocytosis. Viral RNA is then released into the cytoplasm, enters the nucleus, and is translated and replicated in order to infect neighboring cells. In the respiratory tract, this leads to cell death and inflammation, leading to some of the previously discussed symptoms. Innate immunity attacks the virus at the site of the infection, while adaptive immunity develops over the coming weeks as B cells produce antibodies and T cells attack the H1N1 virus. (Chat GPT prompt: Explain the biological mechanisms including the anatomy, organ physiology, cell and molecular physiology of the Swine Flu virus).

Dataset:

(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.)

- The data is from the World Health Organization, however, the WHO page linked on the Kaggle website gives an error. Therefore, information regarding the dataset must be taken from the Kaggle website. The citation for the website is Devakumar, K. P. (n.d.). H1N1 | 2009 | Swine flu pandemic. Kaggle: Your Machine Learning and Data Science Community. The data was based on WHO reporting, which collected confirmed case data through laboratory testing and reports from hospitals. Under-reporting should be considered in the data set as it is often a limitation with public health disease surveillance systems. The user from Kaggle cleaned the data to sort by country. The data is measure in cumulative cases and days.

Data Analyis:

Methods

First, we converted cumultative data to susceptible-infected-recovered data based on the population of Mexico at the time and the infectios period. We estimated β and γ by running an SIR model with the Forward Euler method and performing a grid search to find the β–γ pair that minimizes the sum of squared errors between model-predicted I(t) and observed data. It then uses the RUnge-Kutta method for a higher predicting accuracy. Two tests were done, the first where the model was trained on the entire data set and the second where the model was trained on only the first half of the data set. In the second case, the first half of the data set was used in order to predict the second half of the data set.

Analysis

- To analyze, we calculated SSE for the data:
- 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 for the first half of the data set were 0.4812 and 0.7, respectively, while the gamma and beta we found for the original data set were 0.2 and 0.5. The SSE calculated for the second half of the data was 244954818879.03757, while the original SSE was 96047308503193.4. The new SSE was lower, likely because the fit was better for the first half of the data.*
- Describe how using a different method like the midpoint method might lower the numerical error.
- Using a method such as the midpoint method could lower the error because Eule's only uses informaton about the slope from the start of each time step, whil the midpoint method estimates the slope at the middle of the interval. This provides a second-order approximation that more accurately predicts the curvature of the solution and lowers the error.

```
# Checkpoint 1
from main_functions import convert_cumulative_to_SIR
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import numpy as np
import numpy as np, pandas as pd
from datetime import datetime, timedelta

import pandas as pd


def convert_cumulative_to_SIR(df, date_col='date',
cumulative_col='cumulative_cases',
                              population=None, infectious_period=8,
recovered_col=None,
                              new_case_col='new_cases', I_col='I_est',
R_col='R_est', S_col='S_est'):
```

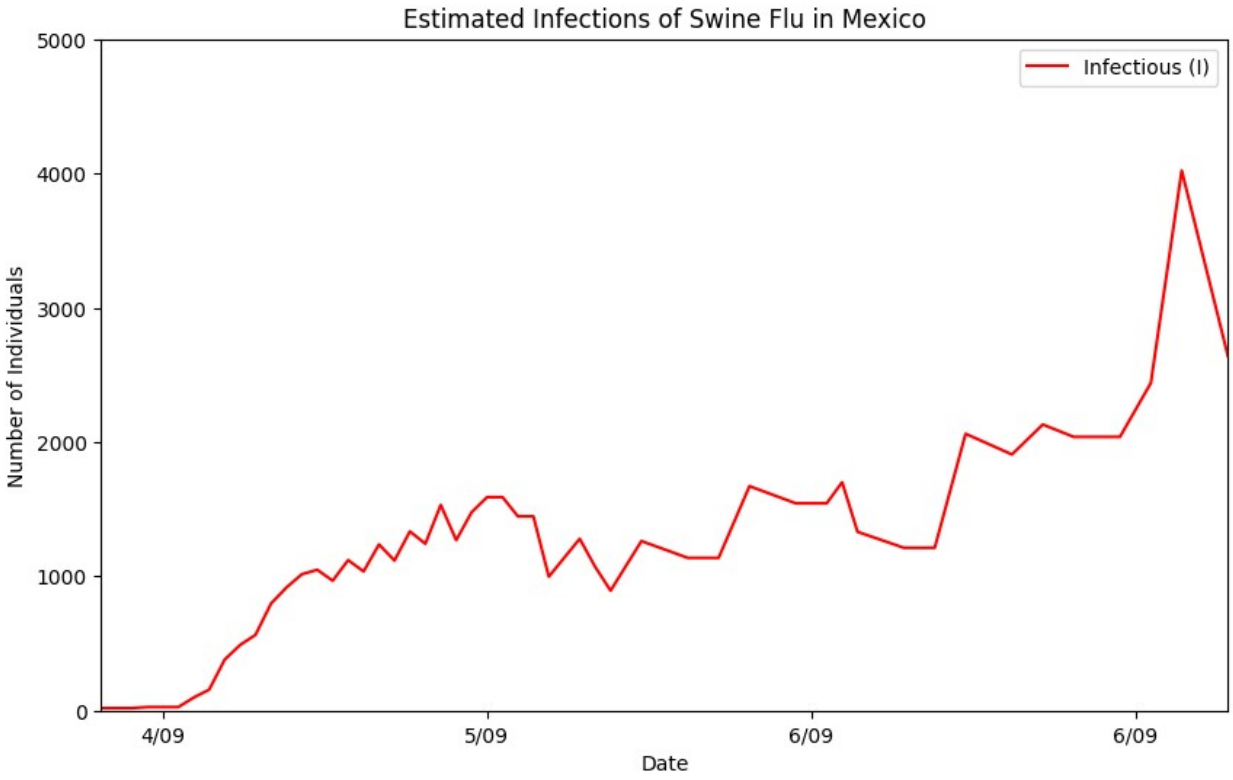
```
    """
    Convert cumulative reported cases into S, I, R estimates for SIR
modeling.
    - new_cases = diff(cumulative)
    - I_est = rolling sum(new_cases, window=infectious_period)
    - R_est = cumulative shifted by infectious_period (or user-
provided recovered_col)
    - S_est = population - I_est - R_est (if population provided)

    Returns a copy of the dataframe with the added columns.
    """
    df = df.copy()
    # Ensure date column sorted if present
    if date_col in df.columns:
        df[date_col] = pd.to_datetime(df[date_col])
        df = df.sort_values(date_col).reset_index(drop=True)

    if cumulative_col not in df.columns:
        raise ValueError(f"Column '{cumulative_col}' not found in
dataframe.")

    # Compute new cases (incident)
    df[new_case_col] = df[cumulative_col].diff().fillna(
        df[cumulative_col].iloc[0])
    df[new_case_col] = df[new_case_col].clip(lower=0)

    # Estimate I(t) as rolling sum over infectious_period
    if infectious_period <= 0:
        raise ValueError("infectious_period must be positive
integer.")
    df[I_col] = df[new_case_col].rolling(
        window=infectious_period, min_periods=1).sum()

    # Estimate R(t)
    if recovered_col and recovered_col in df.columns:
        df[R_col] = df[recovered_col].fillna(0)
    else:
        df[R_col] =
df[cumulative_col].shift(infectious_period).fillna(0)

    # Compute S(t) if population provided
    if population is not None:
        df[S_col] = population - df[I_col] - df[R_col]
        df[S_col] = df[S_col].clip(lower=0)
    else:
        df[S_col] = np.nan

    # Ensure numeric and non-negative
    for col in [new_case_col, I_col, R_col]:
        df[col] = df[col].astype(float).clip(lower=0)
```

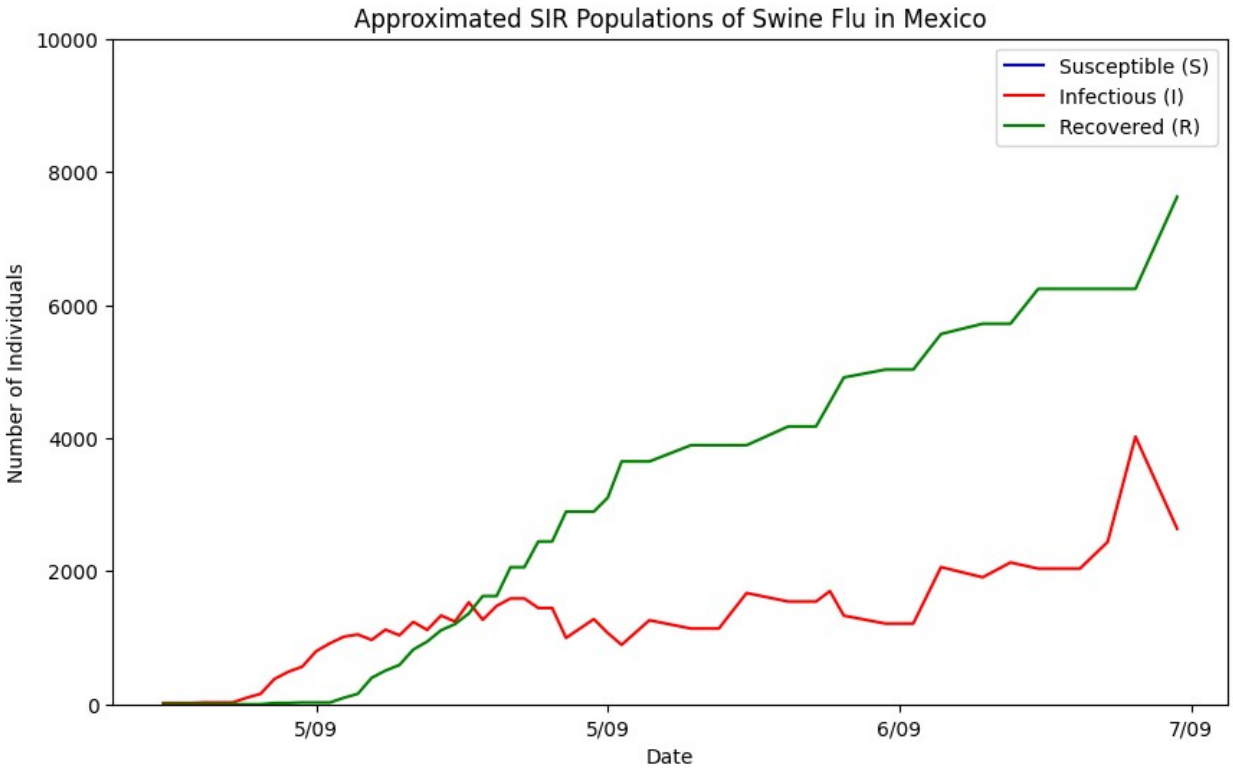
```
    if population is not None:
        df[S_col] = df[S_col].astype(float)

    return df

# Load the Swine flu dataset
data = pd.read_csv('/Users/addisonbuck/Downloads/Module
4/swine_flu_mexico_data_2009_cumulative.csv')
# Display the first few rows of the dataset/Read in the file
print(data.head())
data.columns = ['date', 'confirmed_cases']
data['date'] = pd.to_datetime(data['date'])
# Use SIR function
population = 111000000  # Mexico population approx. as of 2009
data_sir = convert_cumulative_to_SIR(
    data,
    date_col='date',
    cumulative_col='confirmed_cases',
    population=population,
    infectious_period=7, #infection period for swine flu is approx. 7
days
    new_case_col='new_cases',
    I_col='I_est',
    R_col='R_est',
    S_col='S_est')

print(data_sir['date'].min(), data_sir['date'].max())

# Merge SIR data back into your main data variable
#data = sir_df

# Plot the Infectious population over time
plt.figure(figsize=(10, 6))

plt.plot(data_sir['date'],
         data_sir['I_est'],
         label='Infectious (I)',
         color='red')

plt.xlabel('Date')
plt.xlim(pd.Timestamp('2009-04-24'), pd.Timestamp('2009-07-06'))
plt.ylim(0, 5000)

plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%-m/%-y'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=3))
plt.ylabel('Number of Individuals')
plt.title('Estimated Infections of Swine Flu in Mexico')
plt.legend()
plt.show()
```

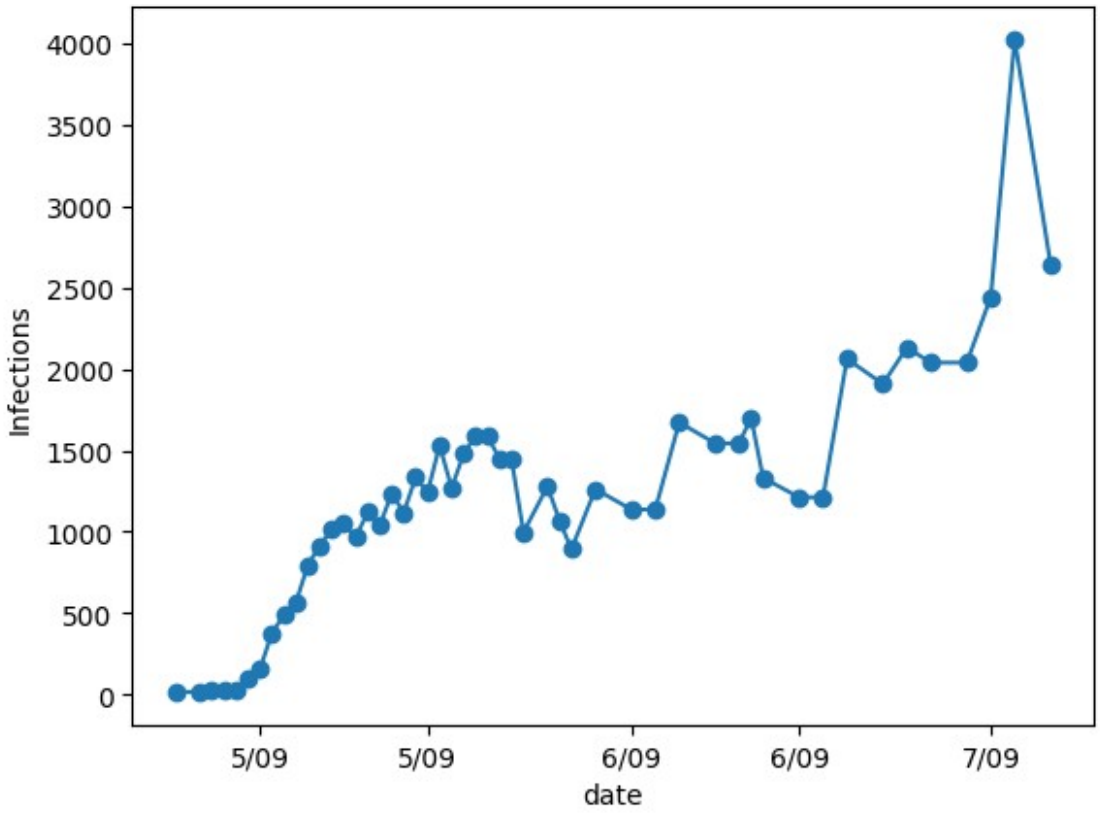
```
# Plot the SIR estimates over time
plt.figure(figsize=(10, 6))
plt.plot(data_sir['date'],
         data_sir['S_est'],
         label='Susceptible (S)',
         color='blue')
plt.plot(data_sir['date'],
         data_sir['I_est'],
         label='Infectious (I)',
         color='red')
plt.plot(data_sir['date'],
         data_sir['R_est'],
         label='Recovered (R)',
         color='green')
plt.xlabel('Date')
plt.ylim(0, 10000)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%-m/%-y'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=3))
plt.ylabel('Number of Individuals')
plt.title('Approximated SIR Populations of Swine Flu in Mexico')
plt.legend()
plt.show()

# Plot partial data

N = 111000000
df_full = pd.read_csv("/Users/addisonbuck/Downloads/Module
4/swine_flu_mexico_data_2009_cumulative.csv")
df_full.columns = ['date', 'confirmed_cases']
df_full['date'] = pd.to_datetime(df_full['date'])

# Run conversion again
df_full = convert_cumulative_to_SIR(
    df_full,
    date_col='date',
    cumulative_col='confirmed_cases',
    population=N,
    infectious_period=7,
    new_case_col='new_cases',
    I_col='I_est',
    R_col='R_est',
    S_col='S_est'
)

# Partial graphs for S, I , R
plt.plot(df_full['date'], df_full['I_est'], 'o-')
plt.xlabel('date'); plt.ylabel('Infections')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%-m/%-y'))
plt.show()
```

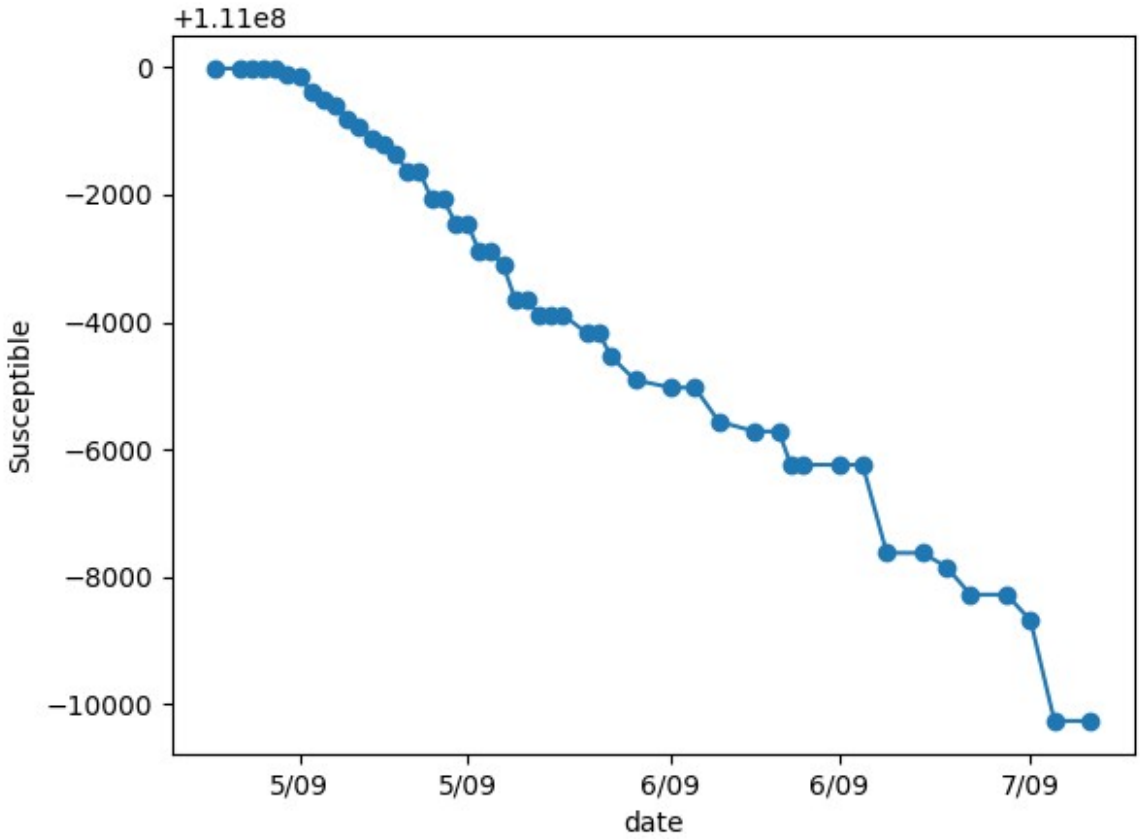
```
plt.plot(df_full['date'], df_full['S_est'], 'o-')
plt.xlabel('date'); plt.ylabel('Susceptible')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%-m/%-y'))
plt.show()

plt.plot(df_full['date'], df_full['R_est'], 'o-')
plt.xlabel('date'); plt.ylabel('Recovered')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%-m/%-y'))
plt.show()
```

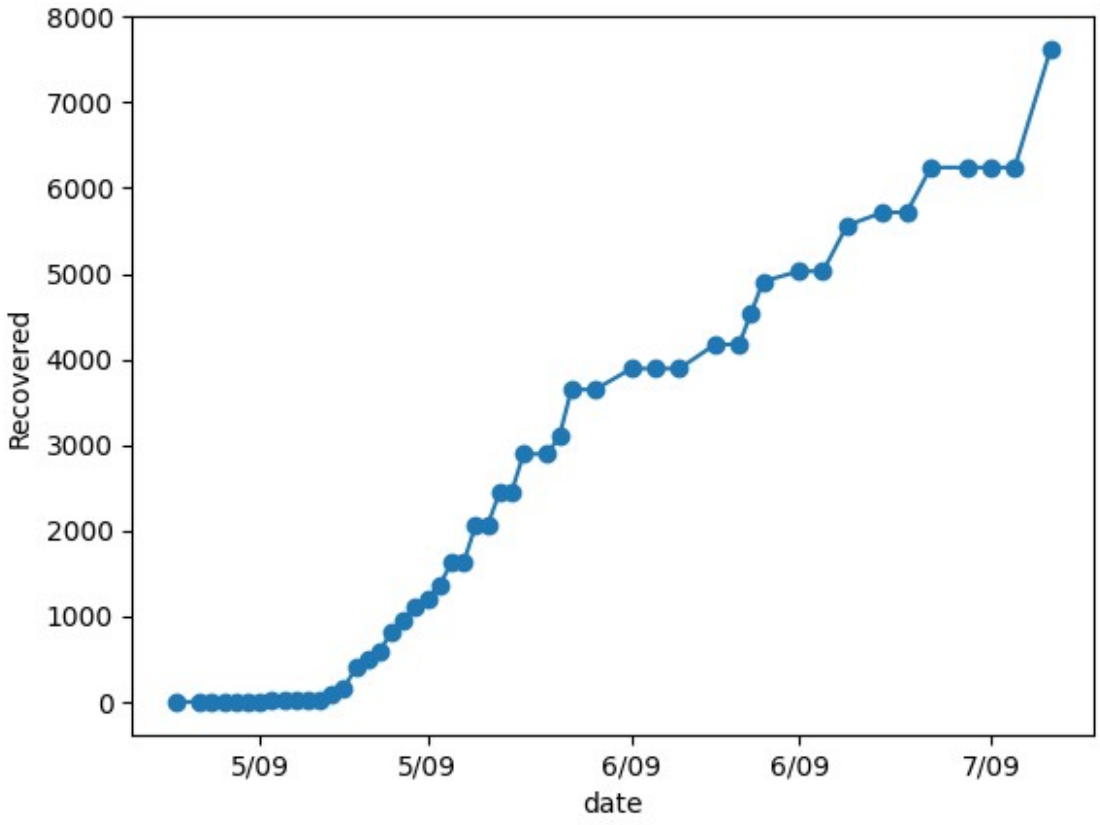
```
         date  confirmed_cases
0  2009-04-24               18
1  2009-04-26               18
2  2009-04-27               26
3  2009-04-28               26
4  2009-04-29               26
2009-04-24 00:00:00 2009-07-06 00:00:00
```




Approximated SIR Populations of Swine Flu in Mexico
Susceptible (S)
Infectious (I)
Recovered (R)
Number of Individuals
10000
8000
6000
4000
2000
0
5/09
5/09
6/09
7/09
Date


Infections
4000
3500
3000
2500
2000
1500
1000
500
0
5/09
5/09
6/09
6/09
7/09
date


+1.11e8
0
−2000
−4000
−6000
−8000
−10000
Susceptible
5/09
5/09
6/09
6/09
7/09
date




```
# Checkpoint 2
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from scipy.integrate import solve_ivp

# ============================================================
# 1. Helper: convert cumulative cases -> S, I, R estimates
# ============================================================

def convert_cumulative_to_SIR(
    df,
    date_col='date',
    cumulative_col='cumulative_cases',
    population=None,
    infectious_period=8,
    recovered_col=None,
    new_case_col='new_cases',
    I_col='I_est',
    R_col='R_est',
    S_col='S_est'
):
    """Convert cumulative reported cases into S, I, R estimates."""
```

```
    df = df.copy()

    # Ensure date column is datetime and sorted
    if date_col in df.columns:
        df[date_col] = pd.to_datetime(df[date_col])
        df = df.sort_values(date_col).reset_index(drop=True)

    if cumulative_col not in df.columns:
        raise ValueError(f"Column '{cumulative_col}' not found in
dataframe.")

    # New daily cases
    df[new_case_col] =
df[cumulative_col].diff().fillna(df[cumulative_col].iloc[0])
    df[new_case_col] = df[new_case_col].clip(lower=0)

    # I(t): rolling sum of new cases over infectious_period
    if infectious_period <= 0:
        raise ValueError("infectious_period must be a positive
integer.")
    df[I_col] = df[new_case_col].rolling(window=infectious_period,
min_periods=1).sum()

    # R(t): shifted cumulative or user-provided
    if recovered_col and recovered_col in df.columns:
        df[R_col] = df[recovered_col].fillna(0)
    else:
        df[R_col] =
df[cumulative_col].shift(infectious_period).fillna(0)

    # S(t): remaining population
    if population is not None:
        df[S_col] = population - df[I_col] - df[R_col]
        df[S_col] = df[S_col].clip(lower=0)
    else:
        df[S_col] = np.nan

    # Clean up types
    for col in [new_case_col, I_col, R_col]:
        df[col] = df[col].astype(float).clip(lower=0)
    if population is not None:
        df[S_col] = df[S_col].astype(float)

    return df


# ============================================================
# 2. Load data (CUMULATIVE) and build SIR estimates
# ============================================================
```

```
# >>> CHANGE THIS PATH IF YOUR FILE IS SOMEWHERE ELSE <<<
csv_path = r"/Users/addisonbuck/Downloads/Module
4/swine_flu_mexico_data_2009_cumulative.csv"

data = pd.read_csv(csv_path)
print("Raw data head (cumulative cases):")
print(data.head())

# Assume two columns: date, cumulative confirmed cases
data.columns = ['date', 'confirmed_cases']
data['date'] = pd.to_datetime(data['date'])

# Mexico population ~2009
population = 111_000_000

# Convert cumulative -> S, I, R (NOT cumulative)
data_sir = convert_cumulative_to_SIR(
    data,
    date_col='date',
    cumulative_col='confirmed_cases',
    population=population,
    infectious_period=7,  # ~7 day infectious period
    new_case_col='new_cases',
    I_col='I_est',
    R_col='R_est',
    S_col='S_est'
)

print("Date range:", data_sir['date'].min(), "to",
data_sir['date'].max())

# Percent of population for plotting
data_sir['S_pct'] = 100 * data_sir['S_est'] / population
data_sir['I_pct'] = 100 * data_sir['I_est'] / population
data_sir['R_pct'] = 100 * data_sir['R_est'] / population

# ============================================================
# 3. Quick plots of raw data and SIR estimates
# ============================================================

# Plot 1: cumulative confirmed cases
plt.figure(figsize=(10, 6))
plt.plot(data_sir['date'], data_sir['confirmed_cases'], marker='o')
plt.xlabel('Date')
plt.ylabel('Cumulative Confirmed Cases')
plt.title('Cumulative Swine Flu Cases in Mexico (Reported Data)')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m/%d'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=7))
plt.tight_layout()
plt.show()
```

```
# Plot 2: S, I, R in percent
plt.figure(figsize=(10, 6))
plt.plot(data_sir['date'], data_sir['S_pct'], label='Susceptible
(S_est)', color='blue')
plt.plot(data_sir['date'], data_sir['I_pct'], label='Infectious
(I_est)', color='red')
plt.plot(data_sir['date'], data_sir['R_pct'], label='Recovered
(R_est)', color='green')
plt.xlabel('Date')
plt.ylabel('Percent of Population (%)')
plt.title('Approximated SIR Populations (Data-based, NOT cumulative)')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m/%d'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=7))
plt.legend()
plt.tight_layout()
plt.show()

# ============================================================
# 4. Euler SIR integrator
# ============================================================

def euler_SIR(beta, gamma, S0, I0, R0, N, dt, n_steps):
    """Euler's method for the SIR model."""
    S = np.zeros(n_steps)
    I = np.zeros(n_steps)
    R = np.zeros(n_steps)

    S[0] = S0
    I[0] = I0
    R[0] = R0

    for n in range(n_steps - 1):
        dS = -beta * S[n] * I[n] / N
        dI = beta * S[n] * I[n] / N - gamma * I[n]
        dR = gamma * I[n]

        S[n+1] = S[n] + dt * dS
        I[n+1] = I[n] + dt * dI
        R[n+1] = R[n] + dt * dR

    return S, I, R

# Basic settings
N = population
dt = 1.0
n_steps = len(data_sir)
t_full = np.arange(n_steps)

S0 = data_sir['S_est'].iloc[0]
```

```
I0 = data_sir['I_est'].iloc[0]
R0 = data_sir['R_est'].iloc[0]

I_true = data_sir['I_est'].values  # NOT cumulative

# ============================================================
# 5. Part 1: Fit beta, gamma on FULL dataset with Euler
# ============================================================

def sse_I_full_euler(beta, gamma):
    S_model, I_model, R_model = euler_SIR(beta, gamma, S0, I0, R0, N,
dt, n_steps)
    errors = I_model - I_true
    return np.sum(errors**2)

# Parameter grid
NBETA = 25
NGAMMA = 25
beta_vals = np.linspace(0.1, 1.0, NBETA)
gamma_vals = np.linspace(0.05, 0.5, NGAMMA)

best_sse_full = np.inf
best_beta_full = None
best_gamma_full = None

for beta in beta_vals:
    for gamma in gamma_vals:
        sse = sse_I_full_euler(beta, gamma)
        if sse < best_sse_full:
            best_sse_full = sse
            best_beta_full = beta
            best_gamma_full = gamma

print("==== Euler fit on FULL data ====")
print("Best beta (full):", best_beta_full)
print("Best gamma (full):", best_gamma_full)
print("Best SSE on full data (Euler):", best_sse_full)

# Best-fit Euler solution on full data
S_full_euler, I_full_euler, R_full_euler = euler_SIR(
    best_beta_full, best_gamma_full, S0, I0, R0, N, dt, n_steps
)

I_full_euler_pct = 100 * I_full_euler / N

# Plot: model vs data for I(t) (full)
plt.figure(figsize=(10, 6))
plt.plot(data_sir['date'], data_sir['I_pct'], 'o', label='Data I_est
(%)', alpha=0.6)
plt.plot(
```

```
    data_sir['date'],
    I_full_euler_pct,
    '-',
    label=f'Euler model I(t) (%) (beta={best_beta_full:.3f},
gamma={best_gamma_full:.3f})'
)
plt.xlabel('Date')
plt.ylabel('Percent Infectious (%)')
plt.title('Euler SIR Fit to Swine Flu Data (Full Dataset)')
plt.text(0.02, 0.95, f"SSE (full) = {best_sse_full:.2e}",
transform=plt.gca().transAxes,
         verticalalignment='top')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m/%d'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=7))
plt.legend()
plt.tight_layout()
plt.show()

# Also: example single "guess" for beta, gamma
beta_guess = 0.5
gamma_guess = 0.2
S_guess, I_guess, R_guess = euler_SIR(beta_guess, gamma_guess, S0, I0,
R0, N, dt, n_steps)
sse_guess = np.sum((I_guess - I_true)**2)
print("Example guess beta, gamma:", beta_guess, gamma_guess)
print("SSE for guess (Euler):", sse_guess)

# ============================================================
# 6. Part 2: Fit on FIRST HALF, predict the SECOND HALF (Euler)
# ============================================================

mid_idx = n_steps // 2  # split index
dates = data_sir['date'].values

I_first = I_true[:mid_idx]
I_second = I_true[mid_idx:]
t_first = np.arange(mid_idx)

def sse_I_first_half_euler(beta, gamma):
    # simulate only first half
    S_model, I_model, R_model = euler_SIR(beta, gamma, S0, I0, R0, N,
dt, mid_idx)
    errors = I_model - I_first
    return np.sum(errors**2)

best_sse_first = np.inf
best_beta_first = None
best_gamma_first = None

for beta in beta_vals:
```

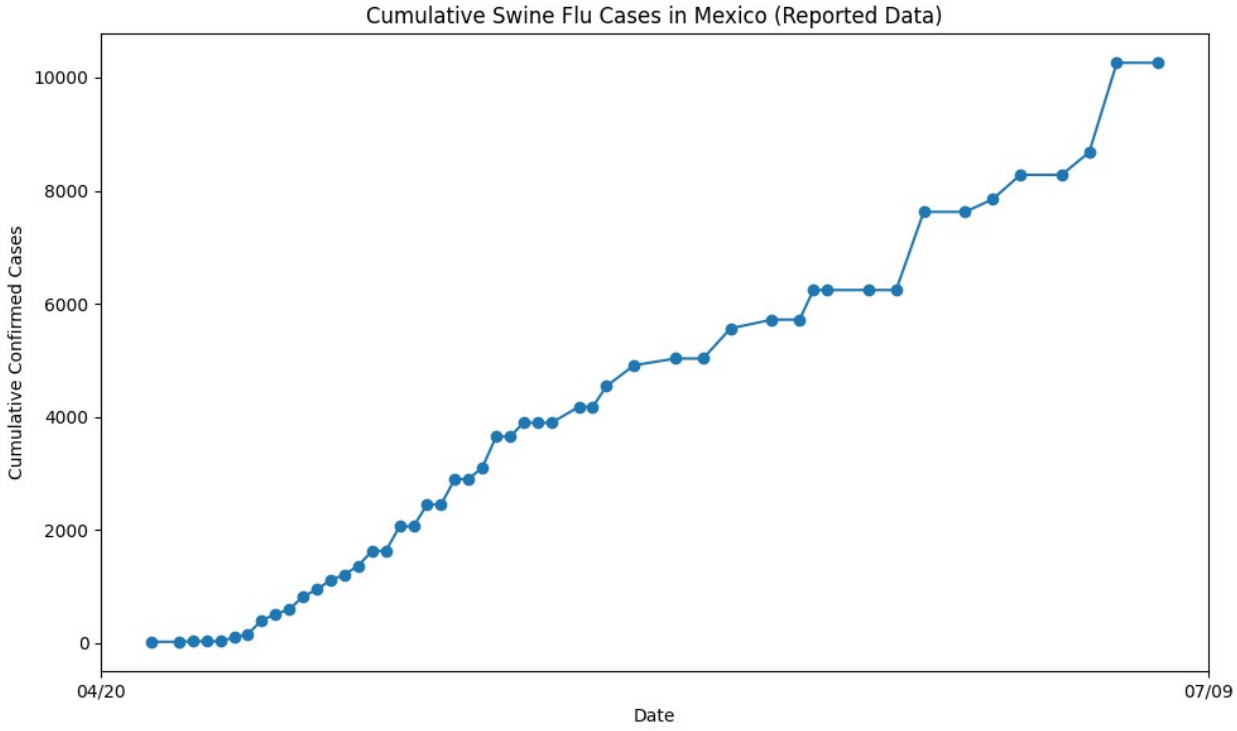
```
    for gamma in gamma_vals:
        sse = sse_I_first_half_euler(beta, gamma)
        if sse < best_sse_first:
            best_sse_first = sse
            best_beta_first = beta
            best_gamma_first = gamma

print("\n==== Euler fit on FIRST HALF of data ====")
print("Best beta (first half, Euler):", best_beta_first)
print("Best gamma (first half, Euler):", best_gamma_first)
print("Best SSE on FIRST HALF (Euler):", best_sse_first)

# Use first-half parameters to simulate the whole time period
S_half_euler, I_half_euler, R_half_euler = euler_SIR(
    best_beta_first, best_gamma_first, S0, I0, R0, N, dt, n_steps
)
I_half_euler_pct = 100 * I_half_euler / N

# SSE on SECOND HALF using first-half parameters
errors_second_half_euler = I_half_euler[mid_idx:] - I_second
sse_second_half_euler = np.sum(errors_second_half_euler**2)

print("SSE on SECOND HALF (Euler, using first-half fit):",
sse_second_half_euler)

# Plot: data vs Euler model from first-half fit, with split line
plt.figure(figsize=(10, 6))
plt.plot(data_sir['date'], data_sir['I_pct'], 'o', label='Data I_est
(%)', alpha=0.6)
plt.plot(
    data_sir['date'],
    I_half_euler_pct,
    '-',
    label=(f'Euler model from FIRST-HALF fit
(beta={best_beta_first:.3f}, '
           f'gamma={best_gamma_first:.3f})')
)
plt.axvline(x=dates[mid_idx], color='gray', linestyle='--',
label='Train/Test Split')
plt.xlabel('Date')
plt.ylabel('Percent Infectious (%)')
plt.title('Euler SIR: Fit on First Half, Predict Full Time Window')
plt.text(
    0.02, 0.95,
    f"SSE (2nd half, Euler) = {sse_second_half_euler:.2e}",
    transform=plt.gca().transAxes,
    verticalalignment='top'
)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m/%d'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=7))
```

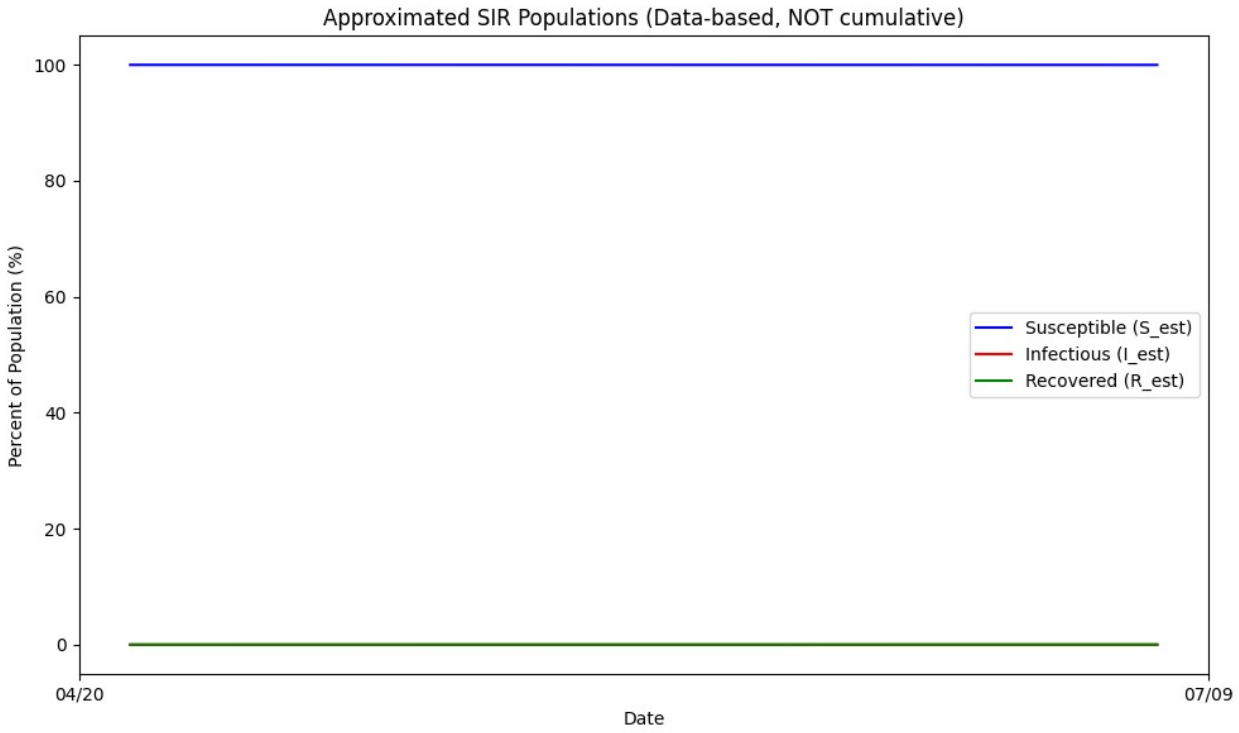
```
plt.legend()
plt.tight_layout()
plt.show()

# ============================================================
# 7. Part 3: RK4 / solve_ivp implementation and fit
# ============================================================

def sir_ode(t, y, beta, gamma, N):
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return [dSdt, dIdt, dRdt]

def solve_ivp_SIR(beta, gamma, S0, I0, R0, N, t_eval):
    """Use scipy.integrate.solve_ivp (Runge-Kutta) to simulate SIR."""
    y0 = [S0, I0, R0]
    sol = solve_ivp(
        fun=lambda t, y: sir_ode(t, y, beta, gamma, N),
        t_span=(t_eval[0], t_eval[-1]),
        y0=y0,
        t_eval=t_eval,
        method='RK45'
    )
    S = sol.y[0]
    I = sol.y[1]
    R = sol.y[2]
    return S, I, R

# Fit beta, gamma on FIRST HALF using RK4 / solve_ivp
def sse_I_first_half_rk(beta, gamma):
    t_eval_first = np.arange(mid_idx)
    S_model, I_model, R_model = solve_ivp_SIR(beta, gamma, S0, I0, R0,
N, t_eval_first)
    errors = I_model - I_first
    return np.sum(errors**2)

best_sse_first_rk = np.inf
best_beta_first_rk = None
best_gamma_first_rk = None

for beta in beta_vals:
    for gamma in gamma_vals:
        sse = sse_I_first_half_rk(beta, gamma)
        if sse < best_sse_first_rk:
            best_sse_first_rk = sse
            best_beta_first_rk = beta
            best_gamma_first_rk = gamma
```

```
print("\n==== RK4 (solve_ivp) fit on FIRST HALF of data ====")
print("Best beta (first half, RK):", best_beta_first_rk)
print("Best gamma (first half, RK):", best_gamma_first_rk)
print("Best SSE on FIRST HALF (RK):", best_sse_first_rk)

# Use RK first-half parameters to simulate the FULL dataset
t_eval_full = np.arange(n_steps)
S_half_rk, I_half_rk, R_half_rk = solve_ivp_SIR(
    best_beta_first_rk, best_gamma_first_rk, S0, I0, R0, N,
t_eval_full
)
I_half_rk_pct = 100 * I_half_rk / N

# SSE on SECOND HALF for RK
errors_second_half_rk = I_half_rk[mid_idx:] - I_second
sse_second_half_rk = np.sum(errors_second_half_rk**2)

print("SSE on SECOND HALF (RK, using first-half fit):",
sse_second_half_rk)

# Plot: data vs RK model from first-half fit
plt.figure(figsize=(10, 6))
plt.plot(data_sir['date'], data_sir['I_pct'], 'o', label='Data I_est
(%)', alpha=0.6)
plt.plot(
    data_sir['date'],
    I_half_rk_pct,
    '-',
    label=(f'RK (solve_ivp) model from FIRST-HALF fit '
           f'(beta={best_beta_first_rk:.3f},
gamma={best_gamma_first_rk:.3f})')
)
plt.axvline(x=dates[mid_idx], color='gray', linestyle='--',
label='Train/Test Split')
plt.xlabel('Date')
plt.ylabel('Percent Infectious (%)')
plt.title('RK4 / solve_ivp SIR: Fit on First Half, Predict Full Time
Window')
plt.text(
    0.02, 0.95,
    f"SSE (2nd half, RK) = {sse_second_half_rk:.2e}",
    transform=plt.gca().transAxes,
    verticalalignment='top'
)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m/%d'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator(interval=7))
plt.legend()
plt.tight_layout()
plt.show()
```

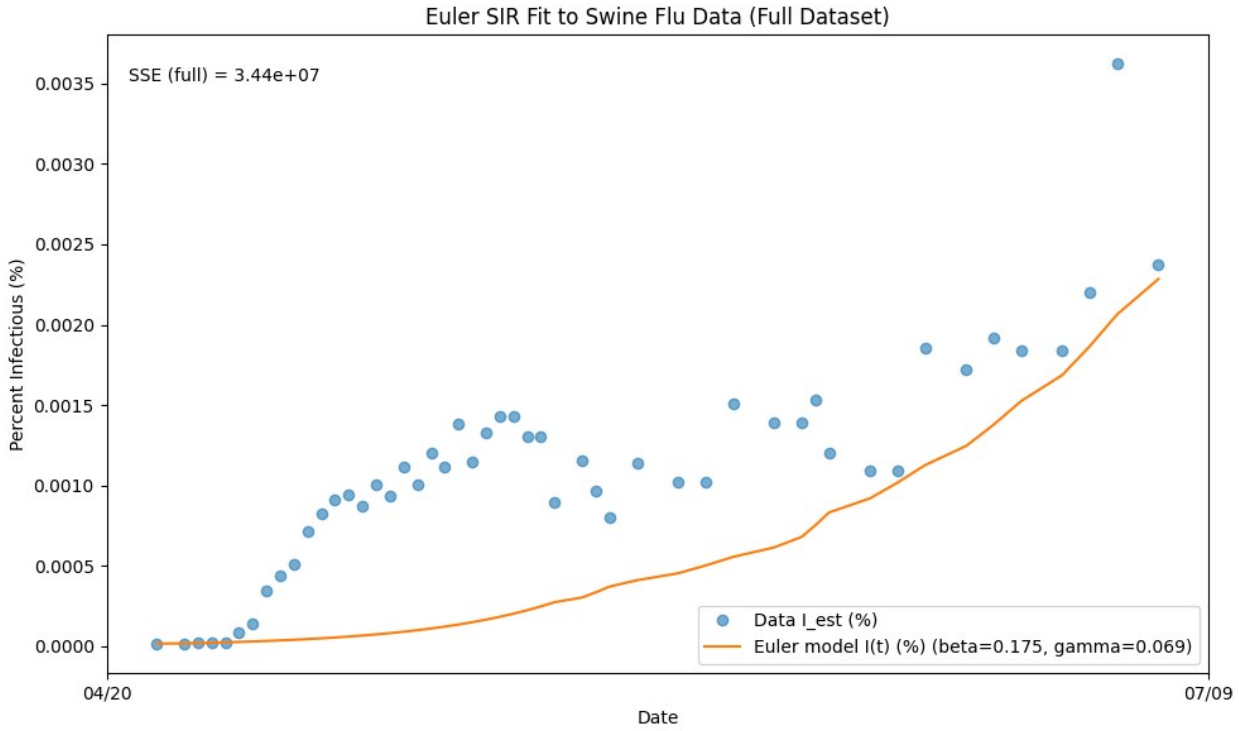
```
# ============================================================
# 8. Print a quick comparison summary
# ============================================================

print("\n==== SUMMARY: SSE on SECOND HALF of data ====")
print(f"Euler (first-half fit) SSE on 2nd half:
{sse_second_half_euler:.3e}")
print(f"RK4   (first-half fit) SSE on 2nd half:
{sse_second_half_rk:.3e}")
```

```
Raw data head (cumulative cases):
         date  confirmed_cases
0  2009-04-24               18
1  2009-04-26               18
2  2009-04-27               26
3  2009-04-28               26
4  2009-04-29               26
Date range: 2009-04-24 00:00:00 to 2009-07-06 00:00:00
```

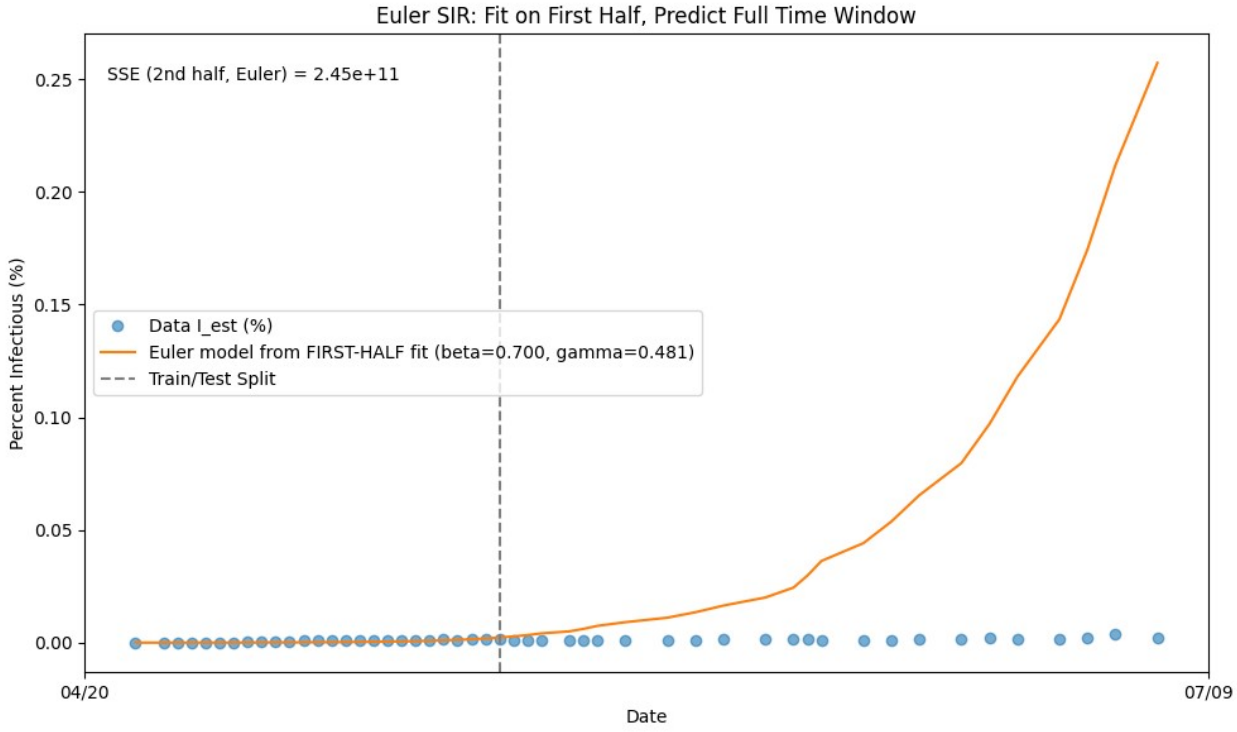




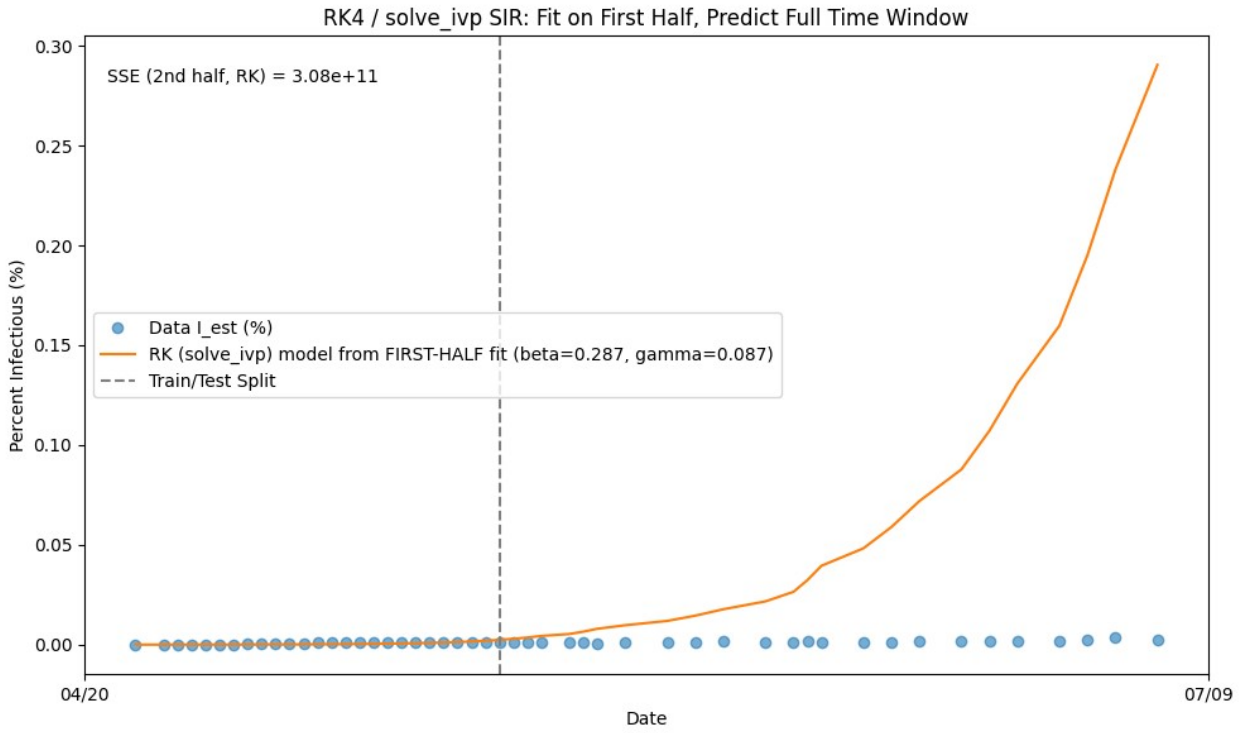

```
==== Euler fit on FULL data ====
Best beta (full): 0.175
Best gamma (full): 0.06875
Best SSE on full data (Euler): 34405810.669
```



```
Example guess beta, gamma: 0.5 0.2
SSE for guess (Euler): 96047308503193.4

==== Euler fit on FIRST HALF of data ====
Best beta (first half, Euler): 0.7
Best gamma (first half, Euler): 0.48124999999999996
Best SSE on FIRST HALF (Euler): 5861587.62423601
SSE on SECOND HALF (Euler, using first-half fit): 244954818879.03757
```




```
==== RK4 (solve_ivp) fit on FIRST HALF of data ====
Best beta (first half, RK): 0.2875
Best gamma (first half, RK): 0.0875
Best SSE on FIRST HALF (RK): 5822409.942381308
SSE on SECOND HALF (RK, using first-half fit): 307946011098.0435
```




```
==== SUMMARY: SSE on SECOND HALF of data ====
Euler (first-half fit) SSE on 2nd half: 2.450e+11
RK4   (first-half fit) SSE on 2nd half: 3.079e+11
```

Verify and validate your analysis:

(Describe how you checked to see that your analysis gave you an answer that you believe (verify). Describe how your determined if your analysis gave you an answer that is supported by other evidence (e.g., a published paper).

Conclusions and Ethical Implications:

(Think about the answer your analysis generated, draw conclusions related to your overarching question, and discuss the ethical implications of your conclusions.

Limitations and Future Work:

(Think about the answer your analysis generated, draw conclusions related to your overarching question, and discuss the ethical implications of your conclusions.

NOTES FROM YOUR TEAM:

- 11/22 Addison updated background info, describing dataset, and graph code.

QUESTIONS FOR YOUR TA:

For the SIR graph, the S (blue) graph is far out of the y range. If we change the Y range to include the S graph, then the I and R graphs flatten out. What should we do to fix this?

How well should the approximation fit the data for the next steps of the project?