Studia i Materiały Informatyki Stosowanej, Tom 15, Nr 2, 2023
str. 10-17

ZASTOSOWANIE JĘZYKA PROGRAMOWANIA PYTHON DO MODELOWANIA ROZPRZESTRZENIANIA SIĘ EPIDEMII

Krzysztof Koziej, Katarzyna Kazimierska-Drobny

Uniwersytet Kazimierza Wielkiego
Wydział Informatyki, Wydział Mechatroniki
ul Kopernika 1, 85-074 Bydgoszcz
e-mail: kkd@ukw.edu.pl

Streszczenie: *Niniejszy artykuł przedstawia numeryczne rozwiązania wybranych modeli rozprzestrzeniania się epidemii w języku programowania Python. Rozwiązania oparto na modelach epidemii SIS, SIR, SIRS oraz SEIR. Do rozwiązań numerycznych w języku Python wykorzystano biblioteki NumPy, SciPy oraz Matplotlib.*

Słowa kluczowe: *Python, modele epidemii, rozwiązania numeryczne, SIS, SIR, SIRS, SEIR*

Application of Python programming language for modeling the evolution of epidemic

Abstarct: *This article presents numerical solutions of selected epidemic spread models in the Python programming language. The solutions were based on the SIS, SIR, SIRS and SEIR epidemic models. NumPy, SciPy and Matplotlib libraries were used for numerical solutions in Python*

Keywords: *Python, models of epidemic, numerical solutions, SIS, SIR, SIRS, SEIR*

1. WSTĘP

Epidemie, pomimo tak wielkiego postępu naukowego, nadal stanowią realne zagrożenie dla zdrowia i życia ludzi i zwierząt. Szczególnie wyraźnie fakt ten widać w ostatnim okresie, gdy pojawiają się kolejne ogniska malarii, ptasiej i świńskiej grypy oraz epidemii SARS. Choroby zakaźne mają bardzo negatywny wpływ na społeczność oraz na wskaźniki ekonomiczne. Medycynie udało się wyeliminować tylko jedną groźną chorobę - ospę prawdziwą. SARS-CoV-2, podobnie jak wirus grypy, wciąż mutuje, liczba wariantów się cały czas zwiększa, więc szanse na to, aby całkowicie go wyeliminować, są nikłe [1,4,10,13,16]. Współczesna epidemiologia chorób zakaźnych wykorzystuje modelowanie matematyczne zarówno do przewidywania rozprzestrzeniania się epidemii, jak i do uzyskania głębszego zrozumienia procesu ewolucji epidemii/pandemii. Badanie chorób zakaźnych jest wysoce interdyscyplinarnym tematem, wymagającym wiedzy z wielu dyscyplin, w szczególności biologicznej wiedzy o patogenie oraz statystycznego opisu dostępnych danych. Przez zastosowanie w epidemiologii modelowania matematycznego możliwe jest przewidywanie scenariuszy rozprzestrzeniania się infekcji w populacjach oraz pozwala na wizualizację dynamiki rozprzestrzeniania się epidemii. Zrozumienie zagrożenia pozwala na systematyczną ewaluację wdrażanych strategii zaradczych poprzez kwarantanny, dystrybucje leków, badania przesiewowe oraz szczepienia [2-3,5,6-7,11,15]. Celem niniejszej pracy jest pracy jest numeryczne rozwiązanie wybranych modeli epidemii w programie Python. Rozwiązania oparto na modelach SIS, SIR, SEIR oraz SIRS.

2. MODELE MATEMATYCZNE EPIDEMII

Do opisu rozprzestrzeniania się epidemii używa się modeli epidemiologicznych. Na ich podstawie można szacować czas osiągnięcia szczytowej liczby zakażeń, liczbę osób hospitalizowanych w danym momencie oraz współczynnik odnowienia choroby. W najprostszych wariantach modele epidemiologiczne są opisane za pomocą równań różniczkowych zwyczajnych, zakładającymi, że osobniki mogą znajdować się w dyskretnej (zależnej od modelu) liczbie stanów oraz są doskonale przemieszane [2,7].

Podstawowe modele epidemiologiczne swoje nazwy zawdzięczają angielskim nazwom stanów, w jakich mogą znaleźć się osobnicy. W najprostszych przypadkach są to klasy: S – osobnicy zdrowi, podatni na infekcje (susceptible), I – osobnicy zarażeni, którzy w wyniku kontaktu z innymi mogą przekazywać chorobę (infetced) oraz R – osobnicy ozdrowiali, którzy nabyli odporność na daną infekcję i nie zarażają (resistant) oraz E – osobnicy narażeni tj. osobnicy zarażeni, ale jeszcze nie zakażeni (exposed) [6]. Dodatkowo modele określają możliwe przejścia między stanami. Najpopularniejsze modele to SIS, SIR, SEIR oraz SIRS, które zostaną omówione w niniejszym artykule. Model SIRS odpowiada sytuacjom, kiedy nie jest możliwe uzyskanie trwałej odporności, zaś w modelu SEIR choroba w początkowym okresie rozwija się w sposób utajony [2-3,5,6-7,11,15].

MODEL SIR

Podstawową cechą modelu jest założenie, iż model ten nie uwzględnia rozrodczości i śmiertelności osobników. Model SIR jest modelem, w którym możliwe jest wyzdrowienie i uzyskanie odporności. Osobnicy ozdrowiali są osobnikami, którzy nie mogą zostać ponownie zarażeni – uzyskali oni trwałą odporność. Dlatego, w ogólności, mogą być także osobnikami, którzy zmarli lub zostali trwale usunięci bądź wyizolowani z populacji. W modelu SIR liczba osobników podatnych maleje na skutek zakażeń. Natomiast liczba osobników zarażonych rośnie ze względu na nowe zakażenia, a maleje ze względu na proces spontanicznych ozdrowień. Liczba osobników ozdrowionych tylko rośnie. Model SIR jest uważany za podstawowy model epidemiologiczny [7,15]. Matematyczną dynamikę zmian liczby zachorowań w badanej populacji w modelu SIR można opisać za pomocą następującego układu równań różniczkowych [11]:

$$\begin{cases} \dfrac{dS}{dt} = -\alpha(t)S(t) \\ \dfrac{dI}{dt} = \alpha I(t)S(t) - \beta I(t) \\ \dfrac{dR}{dt} = \beta I(t) \end{cases} \quad (1)$$

gdzie α oznacza średnią liczbę kontaktów powodujących infekcję a β jest współczynnikiem wyzdrowień (czyli przejścia osobników zainfekowanych po okresie choroby do grupy osobników odpornych na daną infekcję).

MODEL SIS

Założenia modelu epidemii SIS są następujące: liczebność populacji jest stała i wynosi N (rodzi się tyle samo osobników co umiera), NS(t) oznacza liczbę osobników zdrowych, NI(t) - liczba osobników zainfekowanych. Infekcja rozprzestrzenia się przez kontakt, α definiowany jest jako współczynnik urodzin i śmiertelności (z założenia równe), β jest średnią liczbą kontaktów powodująca infekcję, γ jest współczynnikiem wyzdrowień. Liczba zachorowań w populacji jest proporcjonalna do iloczynu obydwu grup i wynosi βNI(t)S(t). Zmiana liczebności obydwu klas w czasie opisana jest za pomocą równań [15]:

$$\begin{cases} \dfrac{d(NS(t))}{dt} = \alpha N - \beta NS(t)I(t) + \gamma NI(t) - \alpha NS(t) \\ \dfrac{d(NI(t))}{dt} = \beta NS(t)I(t) - \gamma NI(t) - \alpha NI(t) \end{cases} \quad (2)$$

MODEL SEIR

SEIR jest modelem stosowanym do modelowania zarażeń m.in. odry, świnki, różyczki. Równania opisujące dynamikę rozwoju epidemii w modelu SEIR mają następującą postać [7,15]:

$$\begin{cases} \dfrac{dS}{dt} = \alpha - r\beta S(t)\dfrac{1}{N} - \mu S(t) \\ \dfrac{dE}{dt} = r\beta S(t)\dfrac{1}{N} - \varepsilon E(t) \\ \dfrac{dI}{dt} = \varepsilon E(t) - \gamma I(t) - \mu I(t) \\ \dfrac{dR}{dt} = \gamma I(t) - \mu R(t) \end{cases} \quad (3)$$

Model SEIR może zawierać parametry zależne od czasu, aby uwzględnić efekty sezonowości, jak również może zawierać dodatkowe przedziały w celu modelowania osób

zaszczepionych i bezobjawowych oraz różnych etapów rozwoju choroby. Może także obejmować wiele grup w celu modelowania heterogeniczności oraz wieku [7].

MODEL SIRS

Model SIRS wykazuje nieco bardziej złożone zachowania niż poprzednie przypadki. Tym razem dynamikę determinują trzy współczynniki (α, β, λ), jednak rozwój epidemii zależy od wartości parametru: α = β / λ. Model SIR zakłada, że po wyzdrowieniu ludzie nabywają na całe życie odporność na daną chorobę; tak jest w przypadku wielu chorób. W przypadku innej klasy chorób przenoszonych drogą powietrzną, na przykład grypy sezonowej, odporność danej osoby może z czasem słabnąć. W tym przypadku stosuje się model SIRS, dzięki któremu osoby, które wyzdrowiały, wracają do stanu podatności. Jeśli istnieje wystarczający napływ do populacji podatnej, w stanie równowagi dynamika będzie w stanie endemicznym z tłumioną oscylacją. Układ równań modelu SIRS ma postać [15]:

$$\begin{cases} \dfrac{dS}{dt} = -\beta \dfrac{S(t)I(t)}{N(t)} + \varepsilon R(t) \\ \dfrac{dI}{dt} = \beta \dfrac{S(t)I(t)}{N(t)} - \gamma I(t) \\ \dfrac{dR}{dt} = \gamma I(t) - \varepsilon R(t) \end{cases} \quad (4)$$

3. IMPLEMENTACJA ROZWIĄZAŃ MODELI EPIDEMII W JĘZYKU PYTHON

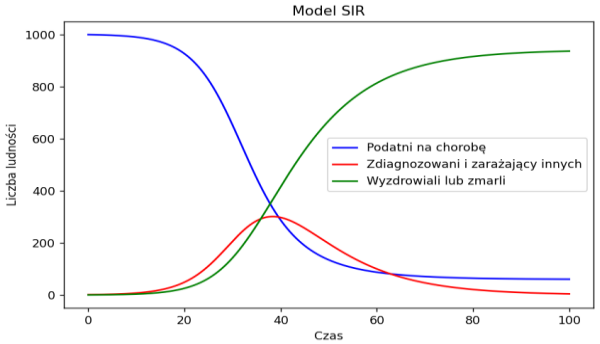
Python to popularny język programowania ogólnego zastosowania, który jest stosowany w wielu dziedzinach, w tym w analizie danych, programowaniu sieciowym, robotyce, grach komputerowych i tworzeniu oprogramowania. Program w języku Python składa się z sekwencji instrukcji, które są wykonywane po kolei. Instrukcje te mogą być przypisaniem wartości do zmiennych, wywołaniem funkcji, instrukcjami warunkowymi, pętlami i innymi [8-9]. Python oferuje wiele narzędzi do analizy danych, takich jak biblioteka Pandas, która pozwala na łatwe przetwarzanie i analizowanie danych tabelarycznych. Do rozwiązywania równań różniczkowych, różnicowych oraz równań algebraicznych można wykorzystać biblioteki, takie jak SciPy i SymPy [17]. Jednymi z bibliotek jakie są wykorzystywane do wizualizacji równań różniczkowych zwyczajnych są NumPy oraz Matplotlib. NumPy to popularna biblioteka Pythona do pracy z wielowymiarowymi tablicami numerycznymi i macierzami, która jest powszechnie stosowana w analizie danych, uczeniu maszynowym i obliczeniach naukowych. Jednym z głównych zalet NumPy jest to, że umożliwia on efektywne wykonywanie operacji na dużych zbiorach danych, co jest kluczowe w analizie danych i obliczeniach naukowych. NumPy jest również często używany z innymi bibliotekami do analizy danych, takimi jak Pandas i Matplotlib [9,12,14,18]. SciPy to biblioteka programistyczna dla języka Python, która dostarcza szereg narzędzi numerycznych i optymalizacyjnych do analizy danych naukowych. SciPy składa się z wielu modułów, z których każdy ma specjalizację w określonej dziedzinie matematyki, takiej jak optymalizacja, analiza danych, równania różniczkowe, transformacja Fouriera, algebra liniowa i wiele innych. SciPy jest często stosowany do rozwiązywania problemów naukowych i inżynierskich, w tym do analizy danych, projektowania układów, symulacji numerycznych, obliczeń matematycznych i innych zadań związanych z nauką i technologią. W skład bilioteki SciPy wchodzi m.in. moduł optimize zawiera wiele funkcji optymalizacyjnych, które umożliwiają minimalizację funkcji jednej lub wielu zmiennych, rozwiązywanie równań nieliniowych oraz wiele innych zadań optymalizacyjnych. Moduł integrate umożliwia numeryczne całkowanie funkcji oraz rozwiązywanie równań różniczkowych zwyczajnych (ODE) i cząstkowych (PDE). SciPy jest jedną z najpopularniejszych bibliotek numerycznych dostępnych w języku Python i jest często wykorzystywana w naukowych i inżynieryjnych projektach programistycznych [9,14,17].
Matplotlib to popularna biblioteka w języku Python do tworzenia wizualizacji i wykresów, która oferuje wiele różnych funkcji do rysowania wykresów, histogramów, powierzchni 3D i innych typów wykresów. Matplotlib jest ważnym narzędziem do wizualizacji danych w Pythonie, które pozwala użytkownikom tworzyć wykresy i wizualizacje w prosty i przyjazny sposób [9,14].

IMPLEMENTACJA MODELU SIR

W celu rozwiązania modelu SIR przyjęto następujące wartości współczynników modelu: N – liczba ludności: 1000; I0 – liczba zakażonych: 1; beta=0; gamma=0.1; S0 – liczba początkowa osobników podatnych: N-I0. Kod rozwiązania modelu SIR ma postać:

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

N = 1000 # całkowita liczba ludności
I0 = 1 # liczba początkowa zakażonych
beta = 0.3 # współczynnik transmisji
gamma = 0.1 # współczynnik usuwania
zakażenia

S0 = N - I0 # liczba początkowa
podatnych
R0 = 0 # liczba początkowa ozdrowieńców

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

t = np.linspace(0, 100, 1000)
y0 = S0, I0, R0
ret = odeint(f, y0, t, args=(N, beta,
gamma))
S, I, R = ret.T

plt.plot(t, S, 'b', label='Podatni na
chorobę) plt.plot(t, I, 'r',
label='Zdiagnozowani i zarażający
innych') plt.plot(t, R, 'g', label=
'Wyzdrowiali lub zmarli')
plt.legend()
plt.xlabel('Czas')
plt.ylabel('Liczba ludności')
plt.title('Model SIR')
plt.show()
```

Funkcja `odeint` z modułu `scipy.integrate` rozwiązuje równania różniczkowe dla zdefiniowanych równań różniczkowych, warunków początkowych i punktów czasowych. Rysunek 1 przedstawia wynik kompilacji powyższego kodu, dla modelu SIR.



Rysunek. 1 Wizualizacja rozwiązania modelu SIR w Pythonie

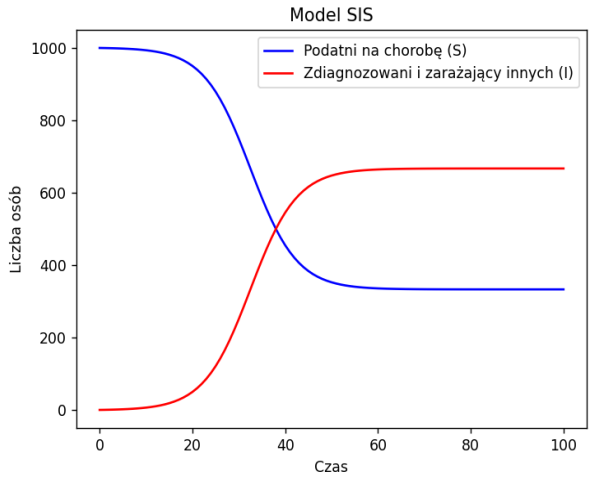
IMPLEMENTACJA MODELU SIS

W celu rozwiązania modelu SIS przyjęto następujące wartości współczynników modelu: N – liczba ludności: 1000; I0 – liczba zakażonych: 1; beta=0.3; gamma=0.1; S0 – liczba początkowa osobników podatnych: N-I0. Kod rozwiązania modelu SIS ma postać:

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# parametry
beta = 0.3  # współczynnik transmisji
gamma = 0.1  # współczynnik wyzdrowień

# warunki początkowe
N = 1000 # Liczba ludzi w populacji
I0 = 1 # Liczba początkowo
zainfekowanych
S0 = N - I0 # Liczba początkowo
podatnych

# czas symulacji
t = np.linspace(0, 100, 101)

# równania różniczkowe
def SIS(y, t, beta, gamma):
    S, I = y
    dS = -beta*S*I / N + gamma*I
    dI = beta*S*I / N - gamma*I
    return dS, dI

# rozwiązanie równań różniczkowych
solution = odeint(SIS, [S0, I0], t,
```

```
args=(beta, gamma))
S, I = solution.T

# wykres
plt.plot(t, S, 'b', label='Podatni na
chorobę (S)')
plt.plot(t, I, 'r', label='Zdiagnozowani
i zarażający innych (I)')
plt.legend()
plt.xlabel('Czas')
plt.ylabel('Liczba osób')
plt.title('Model SIS')
plt.show()
```

Rysunek 2 przedstawia wynik kompilacji powyższego kodu, dla modelu SIS



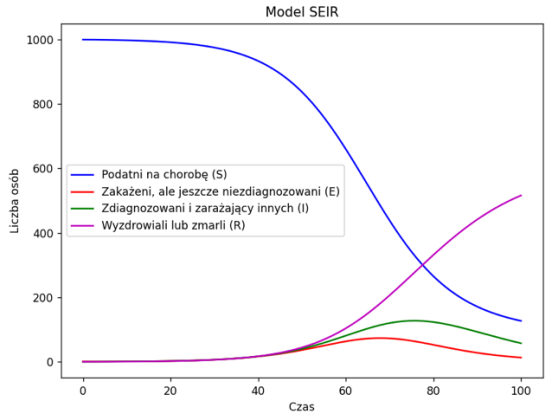
Rysunek. 2 Wizualizacja rozwiązania modelu SIS w Pythonie

IMPLEMENTACJA MODELU SEIR

W celu rozwiązania modelu SEIR przyjęto następujące wartości współczynników modelu: N – liczba ludności: 1000; I0 – liczba osób zdiagnozowanych i zarażających innych: 0; E0 – liczba osób zakażonych, ale jeszcze nie zdiagnozowanych: 1; beta=0.4; gamma=0.1; sigma=0.2; S0 – liczba początkowa osobników podatnych: N-I0; R0 – liczba osób, które wyzdrowiały lub umarły: 0. Kod ro rozwiązania modelu SEIR ma postać:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

N = 1000 # liczba osób w populacji
beta = 0.4 # współczynnik zarażania
gamma = 0.1 # współczynnik wyzdrowienia
sigma = 0.2 # współczynnik przejścia z
grupy E do grupy I

S0 = N - 1 # liczba osób podatnych na
chorobę
E0 = 1 # liczba osób zakażonych, ale
jeszcze niezdiagnozowanych
I0 = 0 # liczba osób zdiagnozowanych i
zarażających innych
R0 = 0 # liczba osób, które wyzdrowiały
lub umarły

t_max = 100
t = np.linspace(0, t_max,
num=t_max*10+1)

def SEIR(y, t, N, beta, gamma, sigma):
    S, E, I, R = y; dSdt = -beta * S * I
/ N; dEdt = beta * S * I / N - (gamma +
sigma) * E; dIdt = sigma * E - gamma *
I; dRdt = gamma * I
    return dSdt, dEdt, dIdt, dRdt

y0 = S0, E0, I0, R0
args = N, beta, gamma, sigma
y = odeint(SEIR, y0, t, args)

plt.plot(t, y[:, 0], 'b', label='Podatni
na chorobę (S)')
plt.plot(t, y[:, 1], 'r',
label='Zakażeni, ale jeszcze
niezdiagnozowani (E)')
plt.plot(t, y[:, 2], 'g',
label='Zdiagnozowani i zarażający innych
(I)')
plt.plot(t, y[:, 3], 'm',
label='Wyzdrowiali lub zmarli (R)')
plt.title('Model SEIR')
plt.xlabel('Czas')
plt.ylabel('Liczba osób')
plt.legend()
plt.show()
```

Rysunek 3 przedstawia wynik kompilacji powyższego kodu, dla modelu SEIR.



Rysunek 3. Wizualizacja rozwiązania modelu SEIR w Pythonie

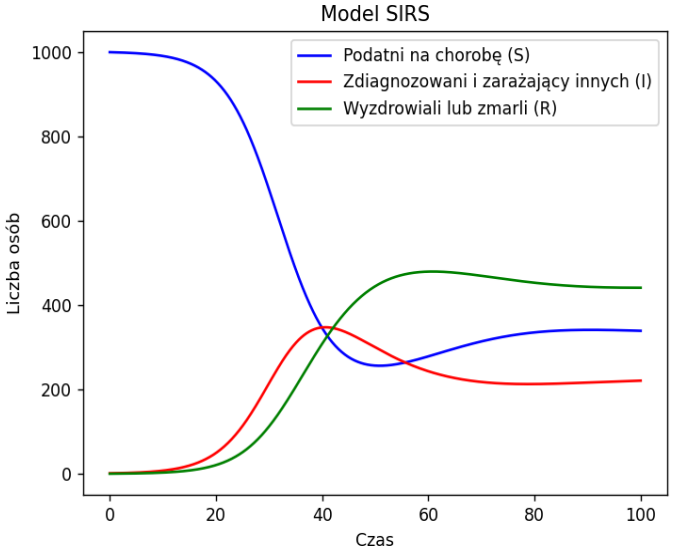
IMPLEMENTACJA MODELU SIRS

W celu rozwiązania modelu SIRS przyjęto następujące wartości współczynników modelu: N – liczba ludności: 1000; I0 – liczba osób zdiagnozowanych i zarażających innych: 1; beta=0.3; gamma=0.1; alpha=0.05; S0 – liczba początkowa osobników podatnych: N-I0-R0; R0 – liczba osób, które wyzdrowiały lub umarły: 0. Kod rozwiązania modelu SIRS ma postać:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

N = 1000 # liczba osób w populacji
beta = 0.3 # współczynnik zarażania
gamma = 0.1 # współczynnik wyzdrowienia
alpha = 0.05 # współczynnik przejścia z
grupy R do grupy S
I0 = 1 # liczba osób zdiagnozowanych i
zarażających innych
R0 = 0 # liczba osób, które wyzdrowiały
lub umarły
S0 = N - I0 - R0 # liczba osób podatnych
na chorobę

def SIRS(y, t, N, beta, gamma, alpha):
    S, I, R = y; dSdt = -beta * S * I /
N + alpha * R; dIdt = beta * S * I / N -
gamma * I; dRdt = gamma * I - alpha * R
    return dSdt, dIdt, dRdt
```

```
t_max = 100
t = np.linspace(0, t_max,
num=t_max*10+1)

y0 = S0, I0, R0
args = N, beta, gamma, alpha
y = odeint(SIRS, y0, t, args)

plt.plot(t, y[:, 0], 'b', label='Podatni
na chorobę (S)')
plt.plot(t, y[:, 1], 'r',
label='Zdiagnozowani i zarażający innych
(I)')
plt.plot(t, y[:, 2], 'g',
label='Wyzdrowiali lub zmarli (R)')
plt.title('Model SIRS')
plt.xlabel('Czas')
plt.ylabel('Liczba osób')
plt.legend()
plt.show()
```

Rysunek 4 przedstawia wynik kompilacji powyższego kodu, dla modelu SIRS.



Rysunek 4. Wizualizacja rozwiązania modelu SIRS w Pythonie

4. PODSUMOWANIE I WNIOSKI

W pracy przedstawiono zastosowanie języka programowania Python do rozwiązywania podstawowych modeli epidemii SIS, SIR, SEIR oraz SIRS. Modelowanie epidemiologiczne jest dobrze rozwiniętą dziedziną badań. Dzięki niemu możliwe jest lepsze zrozumienie mechanizmów rozprzestrzeniania się chorób, identyfikację kluczowych elementów oraz ocena efektywności różnorakich strategii interwencji. W najprostszych wariantach modele epidemiologiczne są opisane równaniami różniczkowymi zwyczajnymi. Język programowania Python może być wykorzystywany w numerycznym rozwiązywaniu takich modeli. Python jest językiem programowania open source, co oznacza, że jest dostępny dla każdego i można go modyfikować i rozwijać zgodnie z potrzebami. Biblioteki Pythona, takie jak NumPy, SciPy, pandas, TensorFlow i PyTorch, są również dostępne na licencjach open source, co oznacza, że są one ogólnodostępne i można je dostosować do konkretnych zastosowań. Python ma dużą i dynamiczną społeczność, która rozwija i utrzymuje różne biblioteki, co oznacza, że są one stale rozwijane i ulepszane. Język ten, jest często stosowany jest w dziedzinie uczenia maszynowego, sztucznej inteligencji i analizie danych, co przyspiesza rozwój bibliotek w tych dziedzinach. Python ma prostą składnię i łatwy w użyciu interfejs, co ułatwia naukę i stosowanie. W porównaniu np. z Matlabem, Python ma bardziej elastyczną składnię, co pozwala na łatwe dostosowywanie kodu do indywidualnych potrzeb i ułatwia tworzenie niestandardowych rozwiązań.

Literatura

1. Anderson R. M., May R. M., Infectious Diseases of Humans: Transmission and Control. Oxford University Press, 1991.
2. Batista A.M., Silvio L.T. de Souza et al., Simulation of deterministic compartmental models for infectious diseases dynamics. arXiv:2106.02085 v1, Cornell University, 2021.
3. Bartłomiejczyk A., Wata M, Analizy epidemiologiczne w środowisku Matlab/Octave, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej 2019, 65, doi: 10.32016/1.65.01.
4. Doshi P., The elusive definition of pandemic influenza. Bull World Health Org, 2012, 89, 532-538.
5. Foryś U., Matematyka w biologii. Wydawnictwo Naukowo-Techniczne, Warszawa, 2005.
6. Foryś U., Dzwonkowska A., Krawczyk J., Matematyka w epidemiologii. Jak modelować covid-19, KOSMOS Problemy nauk biologicznych, 2021, 70 (3), 475–484.
7. Hethcote, H. W., The mathematics of infectious diseases. SIAM review, 42(4), 599-653, 2000.
8. Johansson R., Matematyczny Python, Helion, 2021.
9. Johansson R., Numerical Python: Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib, Helion, 2020.
10. Kiple K.F. (red.), Wielkie epidemie w dziejach ludzkości, Oficyna Wydawnicza Atena, 2002
11. Mata A.S., Dourado S., Mathematical modeling applied to epidemics: an overview. São Paulo Journal of Mathematical Sciences 2023, 15, 1025–1044.
12. McKinney W., Python w analizie danych. Przetwarzanie danych za pomocą pakietów Pandas i NumPy oraz środowiska Jupyter, Helion, 2023.
13. Porta M., A Dictionary of Epidemiology. Oxford University Press, 2008, 81-82.
14. Ramalho L., Zaawansowany Python, APN Promise, 2022.
15. Tadeusiewicz R., Jaworek J i inni. Wprowadzenie do modelowania systemów biologicznych oraz ich symulacji w środowisku MATLAB. Wydawca Uniwersytet Marii Curie-Skłodowskiej w Lublinie, 2012.
16. Report of the review committee on the functioning of the International Health Regulations (2005) and on pandemic influenza A (H1N1). International Health Regulations Review Committee, 2009.
17. Oficjalna strona biblioteki SciPy, https://scipy.org/ Dostęp 06.06.2023.
18. Oficjalna strona biblioteki NumPy, https://numpy.org/ Dostęp 06.06.2023.