Rafał Mokrzycki 101807

Analiza zachorowań na SARS-CoV-2 w Polsce i krajach regionu

Polska, Niemcy, Czechy, Słowacja, Ukraina, Białoruś, Litwa, Łotwa

Wzrost liczebności populacji, w tym także np. liczby zachorowań na jakąś chorobę można przybliżać różnymi funkcjami matematycznymi. Model wykładniczy, najprostszy, pozwala jedynie na oszacowanie fazy początkowej i fazy wzrostu wykładniczego, jednak w następnych okresach zaczyna znacząco odbiegać od rzeczywistych danych. Rzeczywistość znacznie lepiej odzwierciedla model logistyczny oparty na funkcji logistycznej, znanej również pod nazwą fukcji sigmoidalnej. W najprostszej postaci model ten ma następujący wzór:

$$f(x) = \frac{1}{1+e^{-x}}$$

Taki model odwierciedla dane w zakresie wartości od 0 do 1, więc nie jest zbyt użyteczny w modelowaniu wzrostu populacji. Powstało wiele udoskonaleń tego modelu, biorących pod uwagę parametry szybkości wzrostu wykładniczego, nachylenia funkcji w fazie stałego wzrostu, szybkości wzrostu w fazie spowolnienia wzrostu oraz górną asymptotę fukcji. Modele te pozwalają na oszacowanie odległości od punktu t0 do punktu przegięcia oraz do przyjętego apriorycznie punktu tmax.

W poniższych rozważaniach zastosowano trójparametrowy model logistyczny o następującym wzorze funkcji:

$$f(t) = \frac{c}{1+ae^{-bt}}$$

gdzie:

a **- szybkość wejścia w fazę stabilnego wzrostu po fazie wzrostu wykładniczego**

b **- szybkość wzrostu w punkcie przegięcia**

c **- maksymalna liczebność populacji**

t **- czas**

Model dobrze przybliża fazy: początkową, wzrostu wykładniczego i stabilnego wzrostu, nieco gorzej estymuje fazy końcowe epidemii. Po numerycznym ustaleniu parametrów a, b i c i ich późniejszym ręcznym dostosowaniu, model ten może jednak przy zadanym czasie t dość dobrze przybliżyć przyszły wzrost liczby zakażeń SARS-CoV-2.

Import bibliotek

In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pylab
from scipy import special
import scipy.optimize as optim
import seaborn as sn
from collections import Counter
import curveball as cb
from datetime import datetime
from datetime import timedelta
```

Import danych dotyczących wirusa

In [5]:

```
%sql

drop table if exists who_temp;

create table who_temp --dane dot. zachorowań i śmierci na koronawirusa z podziałem na kr
aje i dni
as select Country, Date, Confirmed, Death, newConfirmed, newDeath
from who
where Country in ('Poland', 'Germany',  'Czechia', 'Slovakia', 'Lithuania', 'Latvia', 'U
kraine', 'Belarus'); --wybor Polski i krajow regionu

select *
from who_temp;
```

Country	Date	Confirmed	Death	newConfirmed	newDeath
Belarus	2020-02-28T00:00:00.000+0000	1.0	0.0	1.0	0.0
Belarus	2020-02-29T00:00:00.000+0000	1.0	0.0	0.0	0.0
Belarus	2020-03-01T00:00:00.000+0000	1.0	0.0	0.0	0.0
Belarus	2020-03-02T00:00:00.000+0000	1.0	0.0	0.0	0.0
Belarus	2020-03-03T00:00:00.000+0000	1.0	0.0	0.0	0.0
Belarus	2020-03-04T00:00:00.000+0000	6.0	0.0	5.0	0.0
Belarus	2020-03-05T00:00:00.000+0000	6.0	0.0	0.0	0.0
Belarus	2020-03-06T00:00:00.000+0000	6.0	0.0	0.0	0.0
Belarus	2020-03-07T00:00:00.000+0000	6.0	0.0	0.0	0.0
Belarus	2020-03-08T00:00:00.000+0000	6.0	0.0	0.0	0.0

Import danych pomocniczych dotyczących liczebności populacji

In [7]:

```
%sql

drop table if exists population_temp;

create table population_temp --dane dot. liczebności populacji wszystkich krajów świata
as select Location Country, PopTotal*1000 stablePop
from population
where Time = 2020 and Variant = 'Medium'
and Location in ('Poland', 'Germany',  'Czechia', 'Slovakia', 'Lithuania', 'Latvia', 'Ukr
aine', 'Belarus'); --wybor Polski i krajow regionu

select *
from population_temp;
```

Country	stablePop
Belarus	9449321.0
Poland	3.7846604E7
Latvia	1886202.0
Lithuania	2722291.0
Slovakia	5459643.0
Ukraine	4.3733756E7
Czechia	1.0708982E7
Germany	8.3783944E7

Połączenie obu ramek danych

In [9]:

```
# przypisanie listy krajów regionu do zmiennej
region_countries = ["Poland", "Germany",  "Czechia", "Slovakia", "Lithuania", "Latvia",
"Ukraine", "Belarus"]

# przypisanie tymczasowych baz danych do zmiennej w postaci ramki danych
who = sqlContext.sql("select * from who_temp").toPandas()
pop = sqlContext.sql("select * from population_temp").toPandas()

# połączenie obu ramek danych
df = pd.merge(who, pop, how='outer', on='Country')

# usunięcie braków danych
df.dropna(inplace=True)
df.isna().sum()

# zmiana typu column liczbowych na int64
df["stablePop"] = df["stablePop"].astype("int64")
df["Confirmed"] = df["Confirmed"].astype("int64")
df["Death"] = df["Death"].astype("int64")
df["newConfirmed"] = df["newConfirmed"].astype("int64")
df["newDeath"] = df["newDeath"].astype("int64")

# dodanie kolumny ze śmiertelnością, liczoną jako odsetek zmarłych wśród zakażonych
df["Mortality"] = df["Death"] / df["Confirmed"]
df.head()
```

	Country	Date	Confirmed	Death	newConfirmed	newDeath	stablePop	Mortality
0	Belarus	2020-02-28	1	0	1	0	9449321	0.0
1	Belarus	2020-02-29	1	0	0	0	9449321	0.0
2	Belarus	2020-03-01	1	0	0	0	9449321	0.0
3	Belarus	2020-03-02	1	0	0	0	9449321	0.0
4	Belarus	2020-03-03	1	0	0	0	9449321	0.0

Dodanie kolumn z liczbą potwierdzonych przypadków, zgonów i wyzdrowień na 100000 mieszkańców

In [11]:

```
# wyliczenie aktualnej liczby żyjących przy nierealistycznym założeniu braku urodzeń i br
aku śmierci na inne choroby
df["popAlive"] = df["stablePop"] - df["Death"]

# zmiana typu kolumny
df["popAlive"] = df["popAlive"].astype("int64")

per_number_rate = 100000

df["rateConfirmed"] = (df["Confirmed"]*per_number_rate)/df["stablePop"]
df["rateDeath"] = (df["Death"]*per_number_rate)/df["stablePop"]
df["healthyPop"] = df["popAlive"] - (df["Confirmed"] - df["Death"])
# zmiana kolejności kolumn dla porządku
columns_titles = ["Country","Date",
                  "Confirmed","Death",
                  "newConfirmed","newDeath",
                  "rateConfirmed","rateDeath",
                  "Mortality","stablePop","healthyPop", "popAlive"]
df = df.reindex(columns=columns_titles)

# zmiana typu danych w kolumnie Date, żeby wyświetlały się tylko rok, miesiąc i dzień
df['Date'] = df['Date'].apply(lambda x: x.strftime('%Y-%m-%d'))

df.tail()
```

	Country	Date	Confirmed	Death	newConfirmed	newDeath	rateConfirmed	rateDeath	Mortality	stablePop	healthyPop

	Country	Date	Confirmed	Death	newConfirmed	newDeath	rateConfirmed	rateDeath	Mortality	stablePop	healthyPop
820	Ukraine	2020-06-04	25411	748	588	13	58.103859	1.710349	0.029436	43733756	43708345
821	Ukraine	2020-06-05	25964	762	553	14	59.368329	1.742361	0.029348	43733756	43707792
822	Ukraine	2020-06-06	26514	777	550	15	60.625938	1.776660	0.029305	43733756	43707242
823	Ukraine	2020-06-07	26999	788	485	11	61.734922	1.801812	0.029186	43733756	43706757
824	Ukraine	2020-06-08	26999	788	0	0	61.734922	1.801812	0.029186	43733756	43706757

Ramka danych ze śmiertlenością - dziś

In [13]:

```
mortality = df[["Country","Date","Mortality"]]
mortality = mortality[mortality["Date"] == max(mortality["Date"])]

mortality.sort_values(by = ["Country"], inplace = True, ascending = False)
plt.figure()
plt.barh(y = mortality["Country"], width = mortality["Mortality"])
for i, v in enumerate(mortality["Mortality"]):
    plt.text(v + 0.001, i - .1, str(round((v*100),2))+"%", color = "#1f77b4", fontweight
= 'bold')
plt.title("Śmiertleność wg kraju na dzień {}".format(max(mortality["Date"])))
plt.margins(y = 0.005, x = 0.2)

mortality.sort_values(by = ["Mortality"], inplace = True)
plt.figure()
plt.barh(y = mortality["Country"], width = mortality["Mortality"])
for i, v in enumerate(mortality["Mortality"]):
    plt.text(v + 0.001, i - .1, str(round((v*100),2))+"%", color = "#1f77b4", fontweight
= 'bold')
plt.title("Śmiertleność wg wielkości na dzień {}".format(max(mortality["Date"])))
plt.margins(y = 0.005, x = 0.2)
```

In [14]:

```
# funkcja do przesuwania początków kolumn na górę tabeli, żeby można było narysować wykre
sy o wspólnym początku krzywych dla poszczególnych krajów

def all_start_simultaneously(DataFrame):
    DataFrame = DataFrame.fillna(0)
    for i in DataFrame:
        lista1=[]
        lista2=[]
        for j in DataFrame[i]:
            if j == 0:
                lista2.append(j)
            else:
                lista1.append(j)
        lista3=lista1+lista2
        DataFrame[i]=lista3
    DataFrame = DataFrame.replace(to_replace = 0, value = np.nan)
    return DataFrame
```

Ramka danych z liczbą zachorowań - dziś max (number_confirmed_commonstop)

In [16]:

```
# ramka danych z liczbą zachorowań w poszczególnych dniach tak jak jest to w rzeczywistoś
ci
```

```
number_confirmed_commonstop = df.pivot(index = "Date",
                        columns = "Country",
                        values = "Confirmed")
number_confirmed_commonstop=number_confirmed_commonstop.fillna(0)
```

Ramka danych z odsetkiem zachorowań - dziś max (rate_confirmed_commonstop)

In [18]:

```
# ramka danych z odsetkiem zachorowań w poszczególnych dniach tak jak jest to w rzeczywis
tości
rate_confirmed_commonstop = df.pivot(index = "Date",
                        columns = "Country",
                        values = "rateConfirmed")
rate_confirmed_commonstop=rate_confirmed_commonstop.fillna(0)
```

In [19]:

```
rate_confirmed_commonstop_plot = df[["Country","Date","rateConfirmed"]]
rate_confirmed_commonstop_plot = rate_confirmed_commonstop_plot[rate_confirmed_commonstop
_plot["Date"] == max(rate_confirmed_commonstop_plot["Date"])]

rate_confirmed_commonstop_plot.sort_values(by = ["Country"], inplace = True, ascending =
False)
plt.figure()
plt.barh(y = rate_confirmed_commonstop_plot["Country"], width = rate_confirmed_commonstop
_plot["rateConfirmed"])
for i, v in enumerate(rate_confirmed_commonstop_plot["rateConfirmed"]):
    plt.text(v + 15, i - .1, str(round(v,2)), color = "#1f77b4", fontweight = 'bold')
plt.title("Liczba przypadków na {} osób wg kraju na dzień {}".format(per_number_rate,max(
rate_confirmed_commonstop_plot["Date"])))
plt.margins(y = 0.005, x = 0.2)

rate_confirmed_commonstop_plot.sort_values(by = ["rateConfirmed"], inplace = True)
plt.figure()
plt.barh(y = rate_confirmed_commonstop_plot["Country"], width = rate_confirmed_commonstop
_plot["rateConfirmed"])
for i, v in enumerate(rate_confirmed_commonstop_plot["rateConfirmed"]):
    plt.text(v + 15, i - .1, str(round(v,2)), color = "#1f77b4", fontweight = 'bold')
plt.title("Liczba przypadków na {} osób wg wielkości na dzień {}".format(per_number_rate,
max(mortality["Date"])))
plt.margins(y = 0.005, x = 0.2)
```

ramka danych z liczbą zachorowań - wspólny start (number_confirmed_commonstart)

In [21]:

```
# ramka danych z liczbą zachorowań w poszczególnych dniach tak jakby pandiemia zaczęła si
ę we wszystkich krajach jednocześnie
number_confirmed_commonstart = df.pivot(index = "Date",
                        columns = "Country",
                        values = "Confirmed")
number_confirmed_commonstart = all_start_simultaneously(number_confirmed_commonstart)

number_confirmed_commonstart.reset_index(inplace=True)
number_confirmed_commonstart.drop("Date", inplace=True, axis=1)
dfaux=pd.DataFrame([[0]*number_confirmed_commonstart.shape[1]],
                   columns = list(number_confirmed_commonstop.columns))
number_confirmed_commonstart = pd.concat([dfaux,number_confirmed_commonstart], ignore_ind
ex=True)
```

Ramka danych z odetkiem zachorowań - wspólny start

(rate_confirmed_commonstart)

In [23]:

```
# ramka danych z odsetkiem zachorowań w poszczególnych dniach tak jakby pandiemia zaczęła
się we wszystkich krajach jednocześnie
rate_confirmed_commonstart = df.pivot(index = "Date",
                        columns = "Country",
                        values = "rateConfirmed")
rate_confirmed_commonstart = all_start_simultaneously(rate_confirmed_commonstart)

rate_confirmed_commonstart.reset_index(inplace=True)
rate_confirmed_commonstart.drop("Date", inplace=True, axis=1)
dfaux=pd.DataFrame([[0]*number_confirmed_commonstart.shape[1]],
                   columns = list(rate_confirmed_commonstop.columns))
rate_confirmed_commonstart = pd.concat([dfaux,rate_confirmed_commonstart], ignore_index=
True)

#porownanie liczba zachorowan na 100 000 mieszkancow w Polsce i krajach regionu w kolejny
ch dniach pandemii

plt.figure(figsize=[10,10])
plt.plot(rate_confirmed_commonstart[region_countries])
plt.legend(region_countries)
plt.title("Liczba zachorowań na 100 000 mieszkańców\nw Polsce i krajach regionu\nw kolejn
ych dniach pandemii")
plt.show()
```

Zbudowanie modeli logistycznych dla krajów regionu

In [25]:

```
#numeryczne znalezienie parametrów a, b i c dla krajów regionu

# utworzenie docelowej, na razie pustej ramki danych na parametry modelu i inne wielkości
logistic_coeffs = pd.DataFrame(index = region_countries,
                               columns = ["a","b","c","IPday","IPnumber","NRMSD"])

number_confirmed_commonstart_fit = pd.DataFrame(index = number_confirmed_commonstart.inde
x, columns = number_confirmed_commonstart.columns)

# iteracyjne poszukiwanie parametrów
for numiter in range(1000):
    number_confirmed_commonstart_fit_aux = number_confirmed_commonstart_fit[number_confir
med_commonstart_fit.columns[number_confirmed_commonstart_fit.isna().all(axis = 0)]]
    countries = []
    list_a = []
    list_b = []
    list_c = []
    inflection_point_day = []
    inflection_point_number = []
    NRMSD = []
    for k in number_confirmed_commonstart_fit_aux:
        time = list(range(len(number_confirmed_commonstart[k].dropna())))
        t,a,b,c,p0,cov=[None,None,None,None,None,None]
        def my_logistic(t, a, b, c):
            return c / (1 + a * np.exp(-b * t))
        p0 = np.random.exponential(size = 3)

        x = np.array(time) + 1
        y = np.array(number_confirmed_commonstart[k].dropna())
        try:
            (a,b,c),cov = optim.curve_fit(my_logistic, x, y, p0 = p0)
            if a<=0 or b<=0 or c<=0:
                countries.append(k)
                list_a.append(np.nan)
                list_b.append(np.nan)
                list_c.append(np.nan)
                inflection_point_day.append(np.nan)
```

```
                inflection_point_number.append(np.nan)
                NRMSD.append(np.nan)
            else:
                def my_logistic(t):
                    return c / (1 + a * np.exp(-b * t))
                countries.append(k)
                list_a.append(a)
                list_b.append(b)
                list_c.append(c)
                inflection_point_day.append(np.log(a) / b)
                inflection_point_number.append(c / 2)
                NRMSD.append(np.sqrt(sum((my_logistic(x)-y)**2)/len(x)))
                ml = my_logistic(x)
                ml = np.append(ml, np.repeat(np.nan, number_confirmed_commonstart[k].isn
ull().sum()))
                number_confirmed_commonstart_fit_aux[k] = ml
        except  RuntimeError:
            countries.append(k)
            list_a.append(np.nan)
            list_b.append(np.nan)
            list_c.append(np.nan)
            inflection_point_day.append(np.nan)
            inflection_point_number.append(np.nan)
            NRMSD.append(np.nan)

    logistic_coeffs1 = pd.DataFrame(list(zip(list_a,
                                    list_b,
                                    list_c,
                                    inflection_point_day,
                                    inflection_point_number,
                                    NRMSD)),
                                    index = countries,
                                    columns = ["a","b","c","IPday","IPnumber","NRMSD"])
    logistic_coeffs.update(logistic_coeffs1)
    number_confirmed_commonstart_fit.update(number_confirmed_commonstart_fit_aux)

# pierwszy niezerowy rekord w kolumnie kraju, oznaczający pierwsze zachorowanie
firstInfection = number_confirmed_commonstop.ne(0).idxmax()
firstInfection = firstInfection.rename("firstInfection")

# dołączenie dat z pierwszą infekcją do ramki danych
logistic_coeffs = pd.concat([logistic_coeffs, firstInfection], axis=1, sort=True)

# zmiana formatu wyświetlania dat
logistic_coeffs["firstInfection"] = pd.to_datetime(logistic_coeffs["firstInfection"], fo
rmat = '%Y-%m-%d')
logistic_coeffs["firstInfection"] = logistic_coeffs["firstInfection"].dt.date

# transpozycja ramki danych w celu ułatwienia późniejszej iteracji
logistic_coeffs = logistic_coeffs.T
logistic_coeffs
```

	Belarus	Czechia	Germany	Latvia	Lithuania	Poland	Slovakia	Ukraine
a	831.053	58.9713	3552.51	32.154	153.888	49.8103	86.2478	212.475
b	0.0868549	0.10435	0.115101	0.0865686	0.122332	0.0631867	0.115879	0.0790164
c	51035.6	8787.65	175891	1037.69	1575.27	27447.1	1524.64	27709.6
IPday	77.4014	39.0709	71.0281	40.09	41.1686	61.852	38.4645	67.8192
IPnumber	25517.8	4393.82	87945.5	518.843	787.635	13723.6	762.32	13854.8
NRMSD	825.349	384.765	4558.36	46.9327	68.5619	787.923	34.3639	592.981
firstInfection	2020-02-28	2020-03-01	2020-01-28	2020-03-02	2020-02-28	2020-03-05	2020-03-06	2020-03-03

W powyższej tabeli wiersze a, b i c oznaczają parametry funkcji wyznaczone numerycznie, IPday - numer dnia od początku pandemii, na który przypada punkt przegięcia krzywej, IPnumber - liczba zakażonych w punkcie przegięcia, NRMSD - znormalizowany pierwiastek średniego błędu odchylenia standardowego (miara, za pomocą której możemy porównać ze sobą błąd oszacowania dla każdego kraju), firstInfection - data pierwszego odnotowanego przypadku w danym kraju

odnotowanego przypadku w danym kraju.

Wykresy predykcyjne dla poszczególnych krajów na podstawie wyliczonych parametrów

In [28]:

```
# rysowanie wykresów. Punktowy - rzeczywiste dane, Liniowy - krzywa na podstawie logistic
_coeffs,
# x = czas, y = liczba zakażeń

#stworzenie pomocniczej ramki danych na potrzeby rysowania wykresu (bez pierwszego wiersz
a, który w każdej kolumnie ma wartość 0)
logistic_growth = number_confirmed_commonstart#[1:]
def my_logistic(t, a, b, c):
    return c / (1 + a * np.exp(-b * t))

for country1, country2 in zip(number_confirmed_commonstart, logistic_coeffs):
  periods = 200
  time = pd.date_range(logistic_coeffs[country2]["firstInfection"], periods = periods)
  y1 = logistic_growth[country1].append(pd.Series(np.repeat(np.nan, periods - len(logist
ic_growth[country1]))))
  y2 = my_logistic(np.array(range(0, len(time))),
                   logistic_coeffs[country2]["a"],
                   logistic_coeffs[country2]["b"],
                   logistic_coeffs[country2]["c"])
  plt.figure()
  plt.plot(time, y1)
  plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
  plt.title(country1)
  plt.xticks(rotation='vertical')
  plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")
```

Ręczna poprawa parametrów krzywych logistycznych dla niektórych krajów

Powyższe wykresy przedstawiają nałożone na siebie dwie krzywe - rzeczywistej liczby zachorowań oraz predykcji na podstawie znalezionych iteracyjnie parametrów modelu. Ponieważ nie wszystkie parametry idelanie oddają przebieg pandemii do tej pory, prawdopodobnie będą popełniały bardzo duże błędy w przyszłości, dlatego konieczna jest ręczna poprawa niektórych parametrów dla niektórych krajów na podstawie dotychczasowych danych. Priorytetem jest tutaj maksymalnie dobre dopasowanie nowej krzywej do danych w fazie spowolnienia wzrostu.

In [30]:

```
# pomocnicza ramka danych do zmian parametrów

logistic_coeffs_reviewed = logistic_coeffs.iloc[[0,1,2,6],:]
logistic_coeffs_reviewed
```

	Belarus	Czechia	Germany	Latvia	Lithuania	Poland	Slovakia	Ukraine
a	831.053	58.9713	3552.51	32.154	153.888	49.8103	86.2478	212.475
b	0.0868549	0.10435	0.115101	0.0865686	0.122332	0.0631867	0.115879	0.0790164
c	51035.6	8787.65	175891	1037.69	1575.27	27447.1	1524.64	27709.6
firstInfection	2020-02-28	2020-03-01	2020-01-28	2020-03-02	2020-02-28	2020-03-05	2020-03-06	2020-03-03

In [31]:

```
country1 = "Belarus"
country2 = country1
periods = 200
time = pd.date_range(logistic_coeffs[country2]["firstInfection"], periods = periods)
y1 = logistic_growth[country1].append(pd.Series(np.repeat(np.nan, periods - len(logistic
_growth[country1]))))
```

```
plt.figure(figsize=[10,5])

plt.subplot(121)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"],
                 logistic_coeffs[country2]["b"],
                 logistic_coeffs[country2]["c"])
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " original"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.subplot(122)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"]*0.8,
                 logistic_coeffs[country2]["b"]*0.9,
                 logistic_coeffs[country2]["c"]*1.2)
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " changed"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.show()

logistic_coeffs_reviewed.Belarus.a = logistic_coeffs_reviewed.Belarus.a * 0.8
logistic_coeffs_reviewed.Belarus.b = logistic_coeffs_reviewed.Belarus.b * 0.9
logistic_coeffs_reviewed.Belarus.c = logistic_coeffs_reviewed.Belarus.c * 1.2
```

In [32]:

```
country1 = "Czechia"
country2 = country1
periods = 200
time = pd.date_range(logistic_coeffs[country2]["firstInfection"], periods = periods)
y1 = logistic_growth[country1].append(pd.Series(np.repeat(np.nan, periods - len(logistic
_growth[country1]))))

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

plt.subplot(121)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"],
                 logistic_coeffs[country2]["b"],
                 logistic_coeffs[country2]["c"])
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " original"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.subplot(122)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"]*3,
                 logistic_coeffs[country2]["b"]*.8,
                 logistic_coeffs[country2]["c"]*1.3)
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " changed"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.show()

logistic_coeffs_reviewed.Czechia.a = logistic_coeffs_reviewed.Czechia.a * 3
logistic_coeffs_reviewed.Czechia.b = logistic_coeffs_reviewed.Czechia.b * 0.8
logistic_coeffs_reviewed.Czechia.c = logistic_coeffs_reviewed.Czechia.c * 1.3
```

In [33]:

```
country1 = "Germany"
```

```
country1 = "Germany"
country2 = country1
periods = 200
time = pd.date_range(logistic_coeffs[country2]["firstInfection"], periods = periods)
y1 = logistic_growth[country1].append(pd.Series(np.repeat(np.nan, periods - len(logistic
_growth[country1]))))

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

plt.subplot(121)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"],
                 logistic_coeffs[country2]["b"],
                 logistic_coeffs[country2]["c"])
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " original"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.subplot(122)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"]*0.8,
                 logistic_coeffs[country2]["b"]*0.9,
                 logistic_coeffs[country2]["c"]*1.1)
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " changed"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.show()

logistic_coeffs_reviewed.Germany.a = logistic_coeffs_reviewed.Germany.a * 0.8
logistic_coeffs_reviewed.Germany.b = logistic_coeffs_reviewed.Germany.b * 0.9
logistic_coeffs_reviewed.Germany.c = logistic_coeffs_reviewed.Germany.c * 1.1
```

In [34]:

```
country1 = "Latvia"
country2 = country1
periods = 200
time = pd.date_range(logistic_coeffs[country2]["firstInfection"], periods = periods)
y1 = logistic_growth[country1].append(pd.Series(np.repeat(np.nan, periods - len(logistic
_growth[country1]))))

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

plt.subplot(121)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"],
                 logistic_coeffs[country2]["b"],
                 logistic_coeffs[country2]["c"])
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " original"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.subplot(122)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"]*1.2,
                 logistic_coeffs[country2]["b"]*0.9,
                 logistic_coeffs[country2]["c"]*1.1)
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " changed"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.show()
```

```
logistic_coeffs_reviewed.Latvia.a = logistic_coeffs_reviewed.Latvia.a * 1.2
logistic_coeffs_reviewed.Latvia.b = logistic_coeffs_reviewed.Latvia.b * 0.9
logistic_coeffs_reviewed.Latvia.c = logistic_coeffs_reviewed.Latvia.c * 1.1
```

In [35]:

```
country1 = "Lithuania"
country2 = country1
periods = 200
time = pd.date_range(logistic_coeffs[country2]["firstInfection"], periods = periods)
y1 = logistic_growth[country1].append(pd.Series(np.repeat(np.nan, periods - len(logistic
_growth[country1]))))

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

plt.subplot(121)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"],
                 logistic_coeffs[country2]["b"],
                 logistic_coeffs[country2]["c"])
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " original"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.subplot(122)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"],
                 logistic_coeffs[country2]["b"],
                 logistic_coeffs[country2]["c"]*1.1)
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " changed"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.show()

logistic_coeffs_reviewed.Lithuania.c = logistic_coeffs_reviewed.Lithuania.c * 1.1
```

In [36]:

```
country1 = "Poland"
country2 = country1
periods = 200
time = pd.date_range(logistic_coeffs[country2]["firstInfection"], periods = periods)
y1 = logistic_growth[country1].append(pd.Series(np.repeat(np.nan, periods - len(logistic
_growth[country1]))))

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

plt.subplot(121)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"],
                 logistic_coeffs[country2]["b"],
                 logistic_coeffs[country2]["c"])
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " original"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.subplot(122)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"]*0.9,
                 logistic_coeffs[country2]["b"]*0.9,
                 logistic_coeffs[country2]["c"]*1.2)
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
```

```
plt.title(str(country1 + " changed"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.show()

logistic_coeffs_reviewed.Poland.a = logistic_coeffs_reviewed.Poland.a * 0.9
logistic_coeffs_reviewed.Poland.b = logistic_coeffs_reviewed.Poland.b * 0.9
logistic_coeffs_reviewed.Poland.c = logistic_coeffs_reviewed.Poland.c * 1.2
```

In [37]:

```
country1 = "Ukraine"
country2 = country1
periods = 200
time = pd.date_range(logistic_coeffs[country2]["firstInfection"], periods = periods)
y1 = logistic_growth[country1].append(pd.Series(np.repeat(np.nan, periods - len(logistic
_growth[country1]))))

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

plt.subplot(121)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"],
                 logistic_coeffs[country2]["b"],
                 logistic_coeffs[country2]["c"])
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " original"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.subplot(122)
y2 = my_logistic(np.array(range(0, len(time))),
                 logistic_coeffs[country2]["a"]*0.9,
                 logistic_coeffs[country2]["b"]*0.9,
                 logistic_coeffs[country2]["c"]*1.2)
plt.plot(time, y1)
plt.scatter(time, y2, facecolor='None', edgecolor='r', alpha=0.3)
plt.xticks(rotation='vertical')
plt.title(str(country1 + " changed"))
plt.legend(["rzeczywista", "szacunkowa"], title = "liczba zachorowań")

plt.show()

logistic_coeffs_reviewed.Ukraine.a = logistic_coeffs_reviewed.Ukraine.a * 0.9
logistic_coeffs_reviewed.Ukraine.b = logistic_coeffs_reviewed.Ukraine.b * 0.9
logistic_coeffs_reviewed.Ukraine.c = logistic_coeffs_reviewed.Ukraine.c * 1.2
```

In [38]:

```
# ramka danych z ostatecznymi parametrami modeli do wyliczenia predykcji
logistic_coeffs_reviewed
```

	Belarus	Czechia	Germany	Latvia	Lithuania	Poland	Slovakia	Ukraine
a	664.842	176.914	2842.01	38.5848	153.888	44.8293	86.2478	191.228
b	0.0781694	0.0834801	0.103591	0.0779117	0.122332	0.056868	0.115879	0.0711147
c	61242.7	11423.9	193480	1141.45	1732.8	32936.5	1524.64	33251.5
firstInfection	2020-02-28	2020-03-01	2020-01-28	2020-03-02	2020-02-28	2020-03-05	2020-03-06	2020-03-03

Tabela przedstawia parametry funkcji logistycznej poprawione tak, aby funkcja lepiej oddawała stan faktyczny. Wiersz Mortality określa śmiertelność, liczoną jako stosunek liczby zmarłych do liczby zakażonych w dniu 8 czerwca 2020.

Predykcja czasu wygaśnięcia pandemii, liczby zakażonych i liczby zmarłych do tego dnia

In [41]:

```
# założenie: pandemia jest opanowana, kiedy wartość pierwszej pochodnej krzywej liczby za
chorowań spada poniżej 1

#stworzenie pomocniczej ramki danych ze śmiertelnością dziś
mortality_trans = mortality.T
mortality_trans.columns = mortality_trans.iloc[0]
mortality_trans = mortality_trans.reindex(sorted(mortality_trans.columns), axis=1)
logistic_coeffs_reviewed = logistic_coeffs_reviewed.append(mortality_trans)
logistic_coeffs_reviewed = logistic_coeffs_reviewed.iloc[[0,1,2,3,6],:]

# czas: 250 dni
periods = 250

#listy pomocnicze
duration = []
enddays = []
infected = []
dead = []


# pętla zapisująca obliczenia do list pomocniczych
for country3 in logistic_coeffs_reviewed:
    time = pd.date_range(logistic_coeffs_reviewed[country3]["firstInfection"], periods =
periods)
    y = my_logistic(np.array(range(0, len(time))),
                    logistic_coeffs_reviewed[country3]["a"],
                    logistic_coeffs_reviewed[country3]["b"],
                    logistic_coeffs_reviewed[country3]["c"])
    endday = np.where(np.diff(y, n = 1) < 1)[0][0]
    duration.append(endday)
    enddays.append(logistic_coeffs_reviewed[country3]["firstInfection"] + timedelta(days
= np.ceil(endday)))
    infected.append(np.ceil(logistic_coeffs_reviewed[country3]["c"]))
    dead.append(np.ceil(logistic_coeffs_reviewed[country3]["Mortality"] * logistic_coeff
s_reviewed[country3]["c"]))

summary = pd.DataFrame(
  data = [list(logistic_coeffs_reviewed.loc["firstInfection"]),
          enddays,
          duration,
          infected,
          dead],
  index = ["pierwsze zachorowanie",
          "wygaśnięcie epidemii",
           "czas trwania [dni]",
           "zakażeni",
           "zmarli"],
  columns = ["Białoruś", "Czechy", "Niemcy", "Łotwa", "Litwa", "Polska", "Słowacja", "Uk
raina"]
)

summary
```

	Białoruś	Czechy	Niemcy	Łotwa	Litwa	Polska	Słowacja	Ukraina
pierwsze zachorowanie	2020-02-28	2020-03-01	2020-01-28	2020-03-02	2020-02-28	2020-03-05	2020-03-06	2020-03-03
wygaśnięcie epidemii	2020-09-07	2020-07-23	2020-07-18	2020-06-14	2020-05-23	2020-09-20	2020-05-28	2020-09-02
czas trwania [dni]	192	144	172	104	85	199	83	183
zakażeni	61243	11424	193481	1142	1733	32937	1525	33252
zmarli	339	391	9116	27	73	1462	28	971

Tabela przedstawia datę pierwszego zachorowania na koronawirusa, szacunkową datę wygaśnięcia epidemii, czas trwania epidemii w dniach oraz sumaryczną liczbę zakażonych i zmarłych na końcu pandemii z podziałem na kraje. W Polsce epidemia powinna wygasnąć około 20 września 2020 i potrwa 199 dni, licząc od daty

pierwszego zakażenia. Pochłonie 1462 ofiar śmiertelnych, a prawie 33 tysiące osób zakażą się koronawirusem. Należy zaznaczyć, że pedykcja nie uwzględnia nadchodzącego zmniejszenia dystansu społecznego, tj. otwarcia granic, intensyfikacji przemieszczania się i zwiększonego dostępu do wydarzeń zbiorowych. Przy dalszym znoszeniu zakazów i obostrzeń ze strony rządu należy podejść do danych z powyższej tabeli z rezerwą.

In [43]: