

GEO

05.03.2026

Arian Temouri, Annemike Rörig,
Elina Winkler, Fabian Zirkel,
Julianne Kitzinger

MAPPING



INHALT

ÜBERBLICK: DATENSETS UND DATEIEN

Elina Winkler

IMPORTS

Elina Winkler

AUFGABE 1

Fabian Zirkel

AUFGABE 2A

Arian Temouri

AUFGABE 2B

Julianne Kitzinger

AUFGABE 3

Annemike Rörig

Bild: <https://www.heyundo.de/reise/deutschland-frankfurt-tagescamp-lerne-programmieren-mit-python-mit-unterstuetzung-der-ki-tagescamp-2664>

DATENSET - BUNDESALAENDER.JSON

```
1 {
2   "type": "FeatureCollection",
3   "features": [
4     {
5       "type": "Feature", "id": 0,
6       "properties": { "id": "DE-BW", "name": "Baden-Württemberg", "type": "State" },
7       "geometry": { "type": "MultiPolygon", "coordinates": [ [ [ [ 9.658460243225211, 49.776340484619197 ], [ 9.650967597961369,
8         49.76515197753929 ], [ 9.656839370727539, 49.761451721191406 ], [ 9.640399932861612, 49.750141143798828 ], [ 9.652028083801326,
9         49.742759704589844 ], [ 9.652208328247184, 49.73902893066429 ], [ 9.646539688110352, 49.738990783691406 ], [ 9.646719932556209,
10        49.735260009765739 ], [ 9.652379035949934, 49.735301971435774 ], [ 9.652549743652287, 49.731571197509879 ], [ 9.646888732910156,
11        49.731529235839844 ], [ 9.641078948974609, 49.735221862793196 ], [ 9.641409873962459, 49.727748870849894 ], [ 9.630180358886776,
12        49.727668762206974 ], [ 9.635949134826944, 49.7239911394043082 ], [ 9.641580581665039, 49.724029541015909 ], [ 9.647407531738224,
13        49.720340728759766 ], [ 9.647579193115462, 49.716609954834212 ], [ 9.641920089721907, 49.716571807861328 ], [ 9.642100334167765,
14        49.712841033935661 ], [ 9.636459350585994, 49.712799072265568 ], [ 9.631500244140682, 49.697841644287109 ], [ 9.637128829956055,
15        49.697879791259766 ], [ 9.637469291687069, 49.690422058105582 ], [ 9.64877986907959, 49.690502166748104 ], [ 9.66048812866228,
16        49.683109283447379 ], [ 9.671889305114973, 49.683181762695426 ], [ 9.682978630066088, 49.690700531005916 ], [ 9.681968688964844,
17        49.713058471679744 ], [ 9.704829216003475, 49.713199615478572 ], [ 9.704668998718375, 49.716918945312557 ], [ 9.715788841247615,
18        49.724449157714844 ], [ 9.715949058532942, 49.720722198486328 ], [ 9.721829414367733, 49.717029571533317 ], [ 9.722459793090802,
19        49.702121734619368 ], [ 9.734221458435059, 49.694759368896598 ], [ 9.734710693359659, 49.683601379394645 ], [ 9.740410804748763,
20        49.683631896972656 ], [ 9.757230758666935, 49.69120025634777 ], [ 9.756610870361271, 49.706100463867472 ], [ 9.773440361023177,
21        49.71366193847884 ], [ 9.784719467163256, 49.71472076415959 ], [ 9.790429115295694, 49.717510223388672 ], [ 9.795980802885742,
22        49.7212717514892692 ], [ 9.795839309692326, 49.724979400634822 ], [ 9.807269096374796, 49.725078582763956 ], [ 9.801700221435774,
23        49.721309661865234 ], [ 9.802008628845272, 49.713871002197266 ], [ 9.813429832458553, 49.713958740234375 ], [ 9.813579559326456,
24        49.710239410400391 ], [ 9.825158119201944, 49.706600189208928 ], [ 9.836808286132812, 49.699249267578068 ], [ 9.843020439147949,
25        49.688140869140739 ], [ 9.843460083007926, 49.677001953125 ], [ 9.837759971618823, 49.676948547363281 ], [ 9.832489967346191,
26        49.665760040282847 ], [ 9.832929611206168, 49.654621124267805 ], [ 9.838639259338066, 49.654670715332202 ], [ 9.83878135681158,
27        49.650951385498217 ], [ 9.833080291747799, 49.650901794433821 ], [ 9.83892917633051, 49.647239685058594 ], [ 9.839218139648494,
28        49.639801025390625 ], [ 9.856319427490519, 49.639961242675781 ], [ 9.862169265747355, 49.636291503906534 ], [ 9.862459182739258,
29        49.628871917724837 ], [ 9.874138832092342, 49.621551513671932 ], [ 9.868579864502124, 49.617790222168253 ], [ 9.874558448791788,
30        49.610420227050952 ], [ 9.880399703979776, 49.606769561767635 ], [ 9.880538940429688, 49.603061676025391 ], [ 9.863450050354061,
31        49.602909088134822 ], [ 9.869289398193644, 49.599250793457145 ], [ 9.858168061989917, 49.591739654541129 ], [ 9.84678936004633,
32        49.591640472412109 ], [ 9.847208976745776, 49.58052062988304 ], [ 9.852909088134879, 49.580570220947436 ], [ 9.853330612182731,
33        49.569450378418026 ], [ 9.847640037536735, 49.569400787353629 ], [ 9.847779273987044, 49.565700531005859 ], [ 9.836679458618335,
34        49.558181762695369 ], [ 9.825299263000431, 49.558071136474609 ], [ 9.825579643249796, 49.550659179687614 ], [ 9.837099075317383,
35        49.547069549560604 ], [ 9.84847831726097, 49.547180175781364 ], [ 9.848620414733944, 49.543479919433594 ], [ 9.860129356384505,
36        49.539878845214901 ], [ 9.865819931030273, 49.539939880371207 ], [ 9.871099472045842, 49.551090240478629 ], [ 9.8767900466091838,
37        49.551151275634709 ], [ 9.876650810241756, 49.554851531982479 ], [ 9.870678901672306, 49.562191009521769 ], [ 9.881798744201944,
38        49.569709777832145 ], [ 9.881660461425952, 49.573421478271598 ], [ 9.887349120723145, 49.573471069336165 ], [ 9.892010003662166,
39        49.577220916748161 ], [ 9.898598670959586, 49.577270507812557 ], [ 9.904159545898608, 49.581031799316406 ], [ 9.921249389648722,
40        49.581100572509822 ], [ 9.92138957977312, 49.57746887207054 ], [ 9.915699005127124, 49.577419281005973 ], [ 9.915828704834269,
41        49.573719024658203 ], [ 9.904979705810661, 49.558811187744141 ], [ 9.90511894226097, 49.555110931396598 ], [ 9.922199249267578,
42        49.555271148681697 ], [ 9.92776012420677, 49.559020996093864 ], [ 9.933589935302848, 49.555370330810547 ], [ 9.933719635009822,
43        49.551670074463004 ], [ 9.928158760070801, 49.547920227051009 ], [ 9.933850288391284, 49.547969818115234 ], [ 9.933988571167276,
44        49.544269561767692 ], [ 9.92842960357666, 49.540519714355582 ], [ 9.934378623962459, 49.533180236816406 ], [ 9.929219245910758,
45        49.518341064453352 ], [ 9.923529624939249, 49.518200029296875 ], [ 9.923660278320312, 49.514579772949332 ], [ 9.929478645324707,
46        49.51095199584978 ], [ 9.929600298706339, 49.507251739502067 ], [ 9.935429573059309, 49.503608703613452 ], [ 9.929740905761719,
47        49.503551483154354 ], [ 9.929998397827262, 49.496158599853629 ], [ 9.924578666687296, 49.488719940185661 ], [ 9.942019462585449,
48        49.477809906006087 ], [ 9.959210395811159, 49.474281311035384 ], [ 9.964899063110352, 49.474338531494141 ], [ 9.964769363403377,
49        49.478031158447322 ], [ 9.976148605346964, 49.478141784668082 ], [ 9.981710433959904, 49.481891632080362 ], [ 9.981840133667049,
50        49.478191375732649 ], [ 9.993219375610636, 49.478302001953352 ], [ 9.998778343200684, 49.482051849365348 ], [ 10.00466891784668,
51        49.482101440429915 ], [ 10.010290145874308, 49.478462219238338 ], [ 10.021670341491813, 49.478569030761662 ], [ 10.027239799499455,
52        49.482311240779297 ], [ 10.032560348510799, 49.493431091308878 ], [ 10.038139343261776, 49.497169494629134 ], [ 10.049400329589901,
53        49.500961303710994 ], [ 10.05496978759794, 49.504692077636832 ], [ 10.060669898986873, 49.504741668701286 ], [ 10.060600938415812,
54        49.515861511230696 ], [ 10.060298919677734, 49.515811920166129 ], [ 10.0487709045413, 49.5194091796875 ], [ 10.04865932464611,
55        49.523101006640682 ], [ 10.042959213257006, 49.523052215576286 ], [ 10.048539161682072, 49.526790618896484 ], [ 10.065508842468319,
56        49.530632019043082 ], [ 10.071079254150675, 49.534358978271428 ], [ 10.070828437805176, 49.541751861572379 ], [ 10.088059425354174,
57        49.538188934326172 ], [ 10.093758583068791, 49.538238525390739 ], [ 10.093879699707315, 49.534549713134709 ], [ 10.099819183349837,
58        49.52721023559593 ], [ 10.09424018859886, 49.523479461670036 ], [ 10.088548660278377, 49.523429870605469 ], [ 10.082968711853312,
59        49.519691467285384 ], [ 10.083209037780989, 49.512310020076229 ], [ 10.089150428772086, 49.504989624023665 ], [ 10.09473037719755,
60        49.508720397949219 ], [ 10.111819267273177, 49.508861541748047 ], [ 10.120839764404354, 49.50531005859375 ], [ 10.123579025268839,
61        49.497901916503906 ], [ 10.1178913879338, 49.497848510742188 ], [ 10.106719970703296, 49.490379333496378 ], [ 10.106960296630973,
62        49.483009338379077 ], [ 10.118469238281534, 49.479419708252067 ], [ 10.1185800818176326, 49.475738525390625 ], [ 10.124508857727335,
63        49.46842193603527 ], [ 10.124631881713867, 49.464740753173771 ], [ 10.130438804626522, 49.461109161377067 ], [ 10.124859089875545,
64        49.457370758056641 ], [ 10.107898712158487, 49.453540802002124 ], [ 10.102548599243107, 49.442440032958984 ], [ 10.114049911499308,
65        49.438861846923828 ], [ 10.125438690185831, 49.438961029052848 ], [ 10.131248474121378, 49.435329437255973 ], [ 10.142629623413143,
66        49.435428619384766 ], [ 10.14867019653326, 49.4244384765625 ], [ 10.143088340759391, 49.420711517333984 ], [ 10.148778915405387,
67        49.420761108398722 ], [ 10.160608291626204, 49.406139373779411 ], [ 10.154918670654297, 49.406089782714787 ], [ 10.160830127136458,
68        49.398780822753963 ], [ 10.15515043153354, 49.398731231689737 ], [ 10.17243862152128, 49.391529083251953 ], [ 10.15569019317627,
69        49.380340576171818 ], [ 10.14988899230957, 49.38396072387701 ], [ 10.127190664331055, 49.383750915527457 ], [ 10.115939140319995,
70        49.376281738281364 ], [ 10.127639770508097, 49.365348815918026 ], [ 10.127750396728629, 49.361671447753906 ], [ 10.122050285339526,
71        49.361621856689453 ], [ 10.122150421142635, 49.357940673828352 ], [ 10.1279497146600929, 49.354309082031477 ], [ 10.133650779724348,
72        49.35437017187614 ], [ 10.133849143082161, 49.347011566162223 ], [ 10.128150939941634, 49.346961975097599 ], [ 10.116660118103027,
73        49.350521087646484 ], [ 10.111260414123819, 49.33943176269554 ], [ 10.111349105830818, 49.335762023925952 ], [ 10.117150306701717,
74        49.332130432128006 ], [ 10.122850418091048, 49.332191467285384 ], [ 10.122939109802246, 49.328510284423942 ], [ 10.117240905761719,
```

Erklärung

- **GeoJSON-Datei**

- *FeatureCollection* - *Properties* - *Geometry*
- geografischen Grenzen der Bundesländer
- Speicherung Koordinaten als eigenes Objekt = *Feature*

Benutzung - Aufgabe 1

- Form
- Fläche der Bundesländer
 - Einfärbung *Choroplethenkarte*
- Namen zur Zuordnung

DATENSET - COVID_DE.CSV

Erklärung

- CSV-Datei liefert die Zahlen
 - Zeitreihen-Tabelle
- state = Name des Bundeslandes
- date = Datum der Meldung
- cases = Anzahl gemeldete Fälle
- recovered = Anzahl der Genesenen
- deaths = Anzahl der Todesfälle

Benutzung (*Niedersachsen*) - Aufgabe 2

- **Grafik 1A** - Verhältnis zwischen Todesfällen
- **Grafik 1B** - tägliche Todesfälle
- **Grafik 2** - Corona Wellen

1	state,date,cases,recovered,deaths
2	Baden-Wuerttemberg,2020-03-27,1243,1193,50
3	Baden-Wuerttemberg,2020-03-28,1006,949,57
4	Baden-Wuerttemberg,2020-04-03,1270,1189,81
5	Baden-Wuerttemberg,2020-10-18,540,532,8
6	Baden-Wuerttemberg,2020-10-22,1988,1961,27
7	Baden-Wuerttemberg,2020-10-27,2118,2096,22
8	Baden-Wuerttemberg,2020-10-30,2705,2690,15
9	Baden-Wuerttemberg,2020-11-03,2793,2758,35
10	Baden-Wuerttemberg,2020-11-07,2386,2357,29
11	Baden-Wuerttemberg,2020-11-10,2431,2398,33
12	Baden-Wuerttemberg,2020-11-15,1129,1089,40
13	Baden-Wuerttemberg,2020-11-17,2667,2607,60
14	Baden-Wuerttemberg,2020-11-18,3240,3169,71
15	Baden-Wuerttemberg,2020-11-19,3080,2990,90
16	Baden-Wuerttemberg,2020-11-25,2829,2756,73
17	Baden-Wuerttemberg,2020-12-05,2454,2402,52
18	Baden-Wuerttemberg,2020-12-13,1929,1881,48
19	Baden-Wuerttemberg,2020-12-14,2258,2178,80

DATENSET - COVID_PER_STATE.CSV

Erklärung

- Gesamtsumme Fälle, Genesenen und Todesfälle je Bundesland
- state = Bundesland
- cases = Gesamtzahl der Fälle
- recovered = Gesamtzahl der Genesenen
- deaths = Gesamtzahl der Todesfälle

Benutzung - Aufgabe 1 ; 3

- **Aufgabe 1**
 - Wert *cases* bestimmt Farbe der Karte
- **Aufgabe 3**
 - Wert *recovered* sortiert

1	•state,cases,recovered,deaths
2	Baden-Württemberg,5009778,4965489,18985
3	Bayern,6668169,6607699,28116
4	Berlin,1418585,1405612,5446
5	Brandenburg,1104240,1092273,6438
6	Bremen,301050,298768,956
7	Hamburg,802806,796205,3544
8	Hessen,2887077,2853678,12179
9	Mecklenburg-Vorpommern,706888,700439,2733
10	Niedersachsen,3820165,3777849,13190
11	Nordrhein-Westfalen,7918225,7830806,30862
12	Rheinland-Pfalz,1742703,1723709,6833
13	Saarland,484735,480656,2087
14	Sachsen,1944901,1920590,16744
15	Sachsen-Anhalt,954211,943045,6236
16	Schleswig-Holstein,1167600,1158395,3443
17	Thüringen,878491,868273,8224
18	
19	

IMPORTS

```
import pandas as pd
import folium
import json
import matplotlib.pyplot as plt
import seaborn as sns
```

import pandas **as** pd

- Daten lesen und bearbeiten

import folium

- Kartentool (Erstellen, Bibliothek, ...)

import json

- damit man GeoJSON-Datei lesen kann

import matplotlib.pyplot **as** plt

- für Diagramme und Graphen

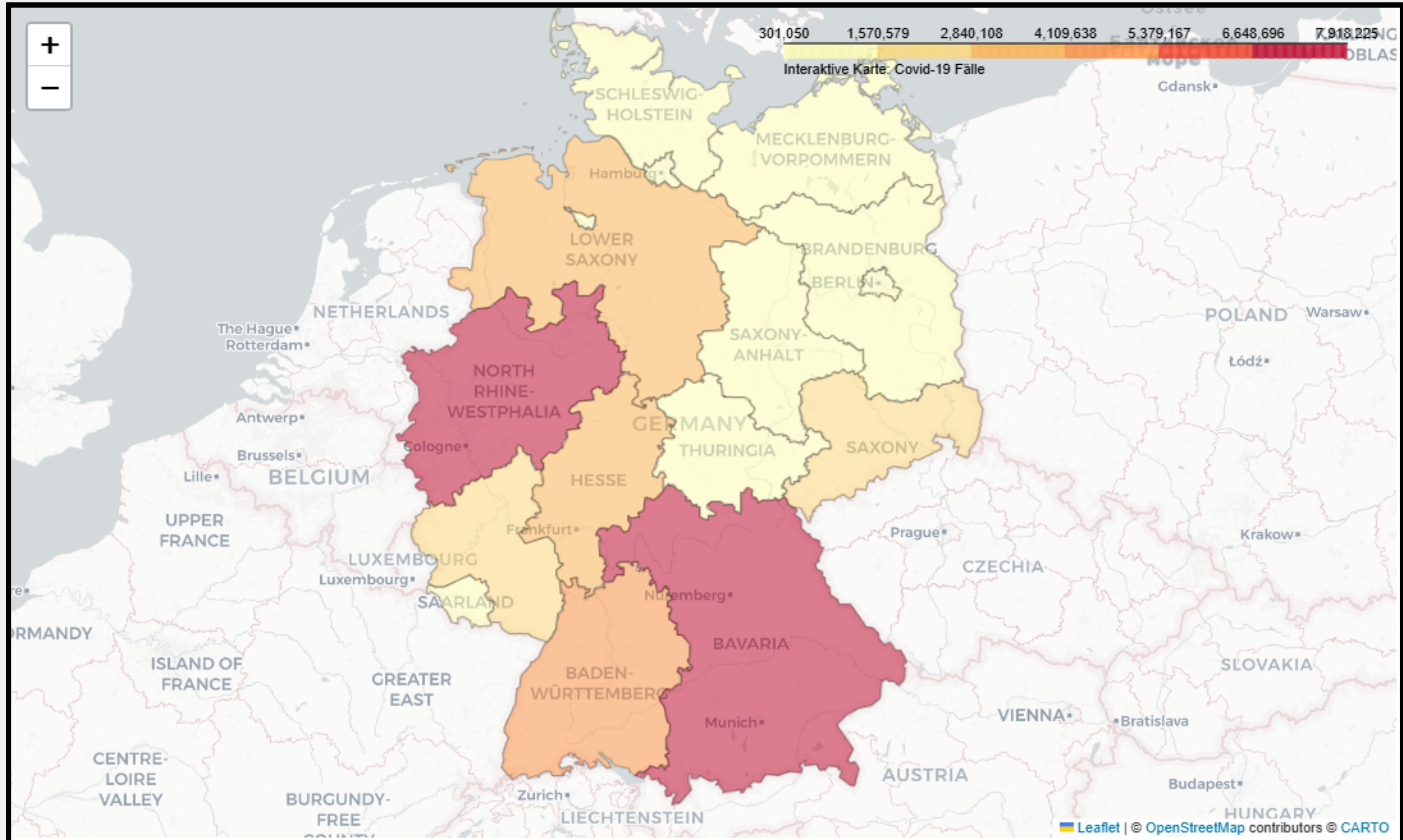
import seaborn **as** sns

- für die Gestaltung von Diagrammen

AUFGABE 1

Verwende die GeoJSON-Datei `bundeslaender.json` und den Datensatz `covid_per_state.csv`, um eine (Folium-)Choroplethenkarte mit den passenden Daten darzustellen.

ERGEBNIS



CODE

```
#CSV Datei einlesen
covid = pd.read_csv("covid_per_state.csv")

#GeoJSON-Datei einlesen
with open ("bundeslaender.json", "r", encoding="utf-8") as f:
    geo_data = json.load(f)

#Karte erstellen
my_map = folium.Map(
    location=[51.163409, 10.447718],
    zoom_start=5.5,
    tiles="CartoDB Positron"
)

#Choroplethenkarte erstellen/hinzufügen
folium.Choropleth(
    geo_data=geo_data,
    data=covid,
    columns=["state", "cases"],
    key_on="feature.properties.name",
    fill_color="YlOrRd",
    fill_opacity=0.5,
    line_opacity=0.3,
    legend_name="Interaktive Karte: Covid-19 Fälle"
).add_to(my_map)

my_map
```

ZUSAMMENFASSUNG

- Darstellung der Covid-Fallzahlen nach Bundesland
- Abhängig von der Fallzahl farblich eingefärbt
- Helle Farben (niedrige Fallzahlen) → dunkle Rote Farben (hohe Fallzahlen)

Direkt sichtbar:

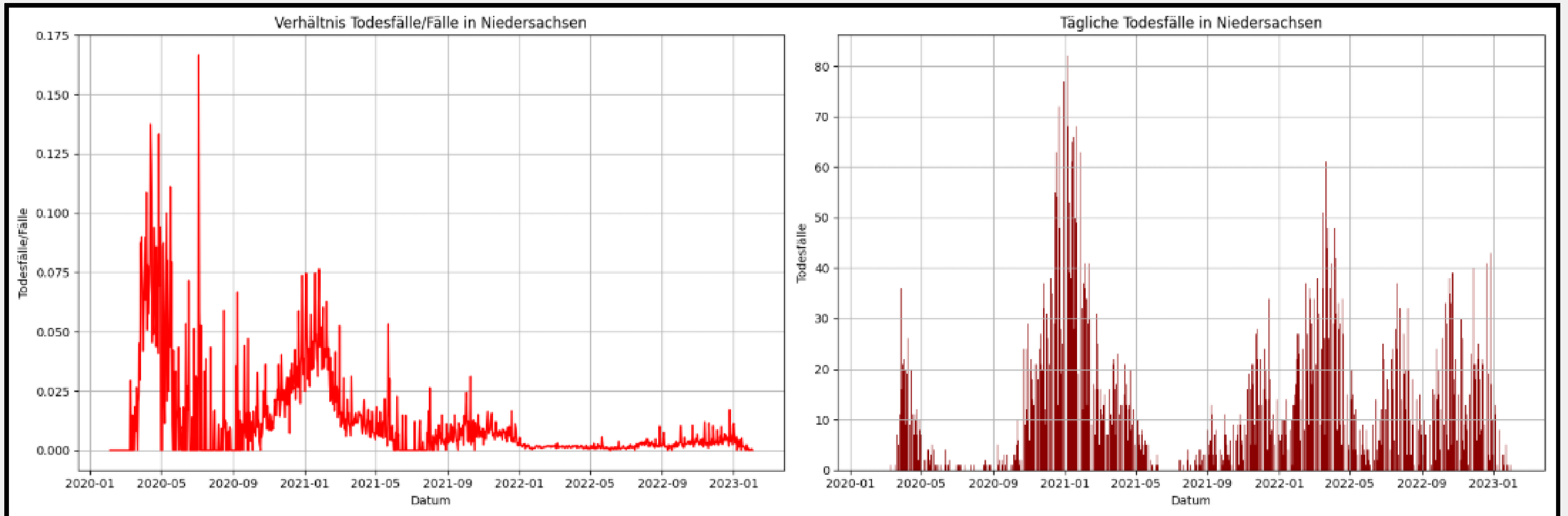
- regionale Unterschiede / räumliche Muster
- Bevölkerungsgröße und Dichte im Zusammenhang mit den Zahlen

→ Karte ermöglicht einen schnellen räumlichen Vergleich / Überblick zwischen den Bundesländern

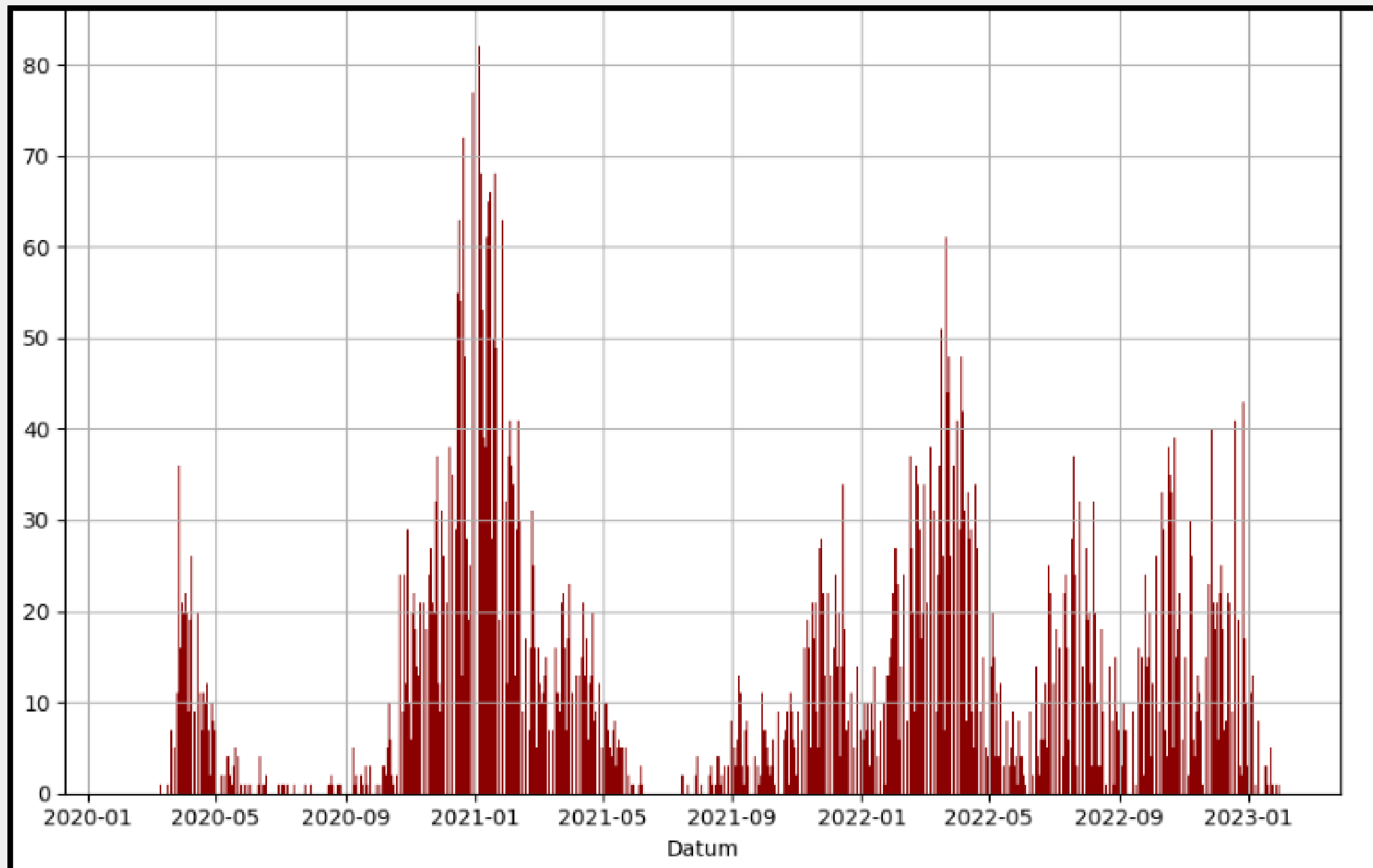
AUFGABE 2A

Zeigen Sie das Verhältnis von Todesfällen zu Infektionen in Niedersachsen und ermitteln Sie den/die Zeitraum /Zeiträume mit den meisten Todesfällen im Zusammenhang mit Covid-19.

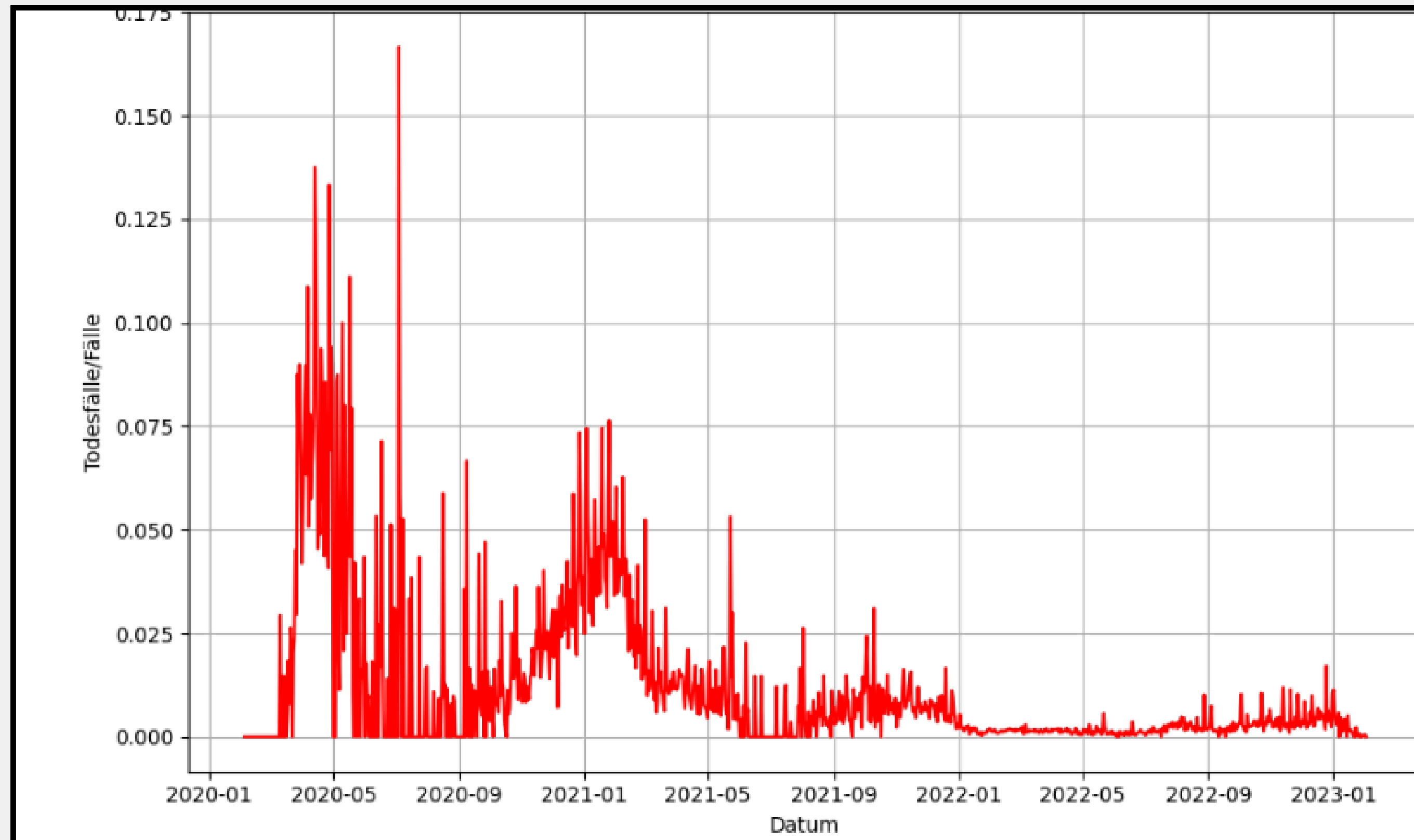
ERGEBNIS (2A)



TÄGLICHE TODESFÄLLE IN NIEDERSACHSEN



VERHÄLTNIS TODESFÄLLE / KRANKHEITSFÄLLE



CODE

```
#CSV-Datei laden und die Spalte "date" als Datumsformat speichern
df = pd.read_csv("covid_de.csv", parse_dates=["date"])

#Nur Zeilen auswählen wo "Niedersachsen" das Bundesland ist
df_nd = df[df["state"] == "Niedersachsen"].copy()

#"Verhältnis Todesfälle zu Fälle berechnen. Formel: "Deaths" / "cases"
df_nd["death_ratio"] = df_nd["deaths"] / df_nd["cases"]

#Daten sortieren nach "Deaths" absteigend
top_deaths = df_nd.sort_values(by="deaths", ascending=False)

#Daten nach chronologischer Reihenfolge sortieren
df_nd = df_nd.sort_values(by='date')

#Diagramm Bereich erstellen, 1 Zeile, 2 Spalten für zwei Diagramme nebeneinander
fig, axes = plt.subplots(1, 2, figsize=(18,6))

#Linkes Diagramm: Todesfälle / Fälle Verhältnis in Niedersachsen
axes[0].plot(df_nd['date'], df_nd['death_ratio'], color='red')
axes[0].set_title('Verhältnis Todesfälle/Fälle in Niedersachsen')
axes[0].set_xlabel('Datum')
axes[0].set_ylabel('Todesfälle/Fälle')
axes[0].grid(True)

#Rechtes Diagramm: Absolute Todesfälle in Niedersachsen
axes[1].bar(df_nd['date'], df_nd['deaths'], color='darkred')
axes[1].set_title('Tägliche Todesfälle in Niedersachsen')
axes[1].set_xlabel('Datum')
axes[1].set_ylabel('Todesfälle')
axes[1].grid(True)

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

ZUSAMMENFASSUNG (2A)

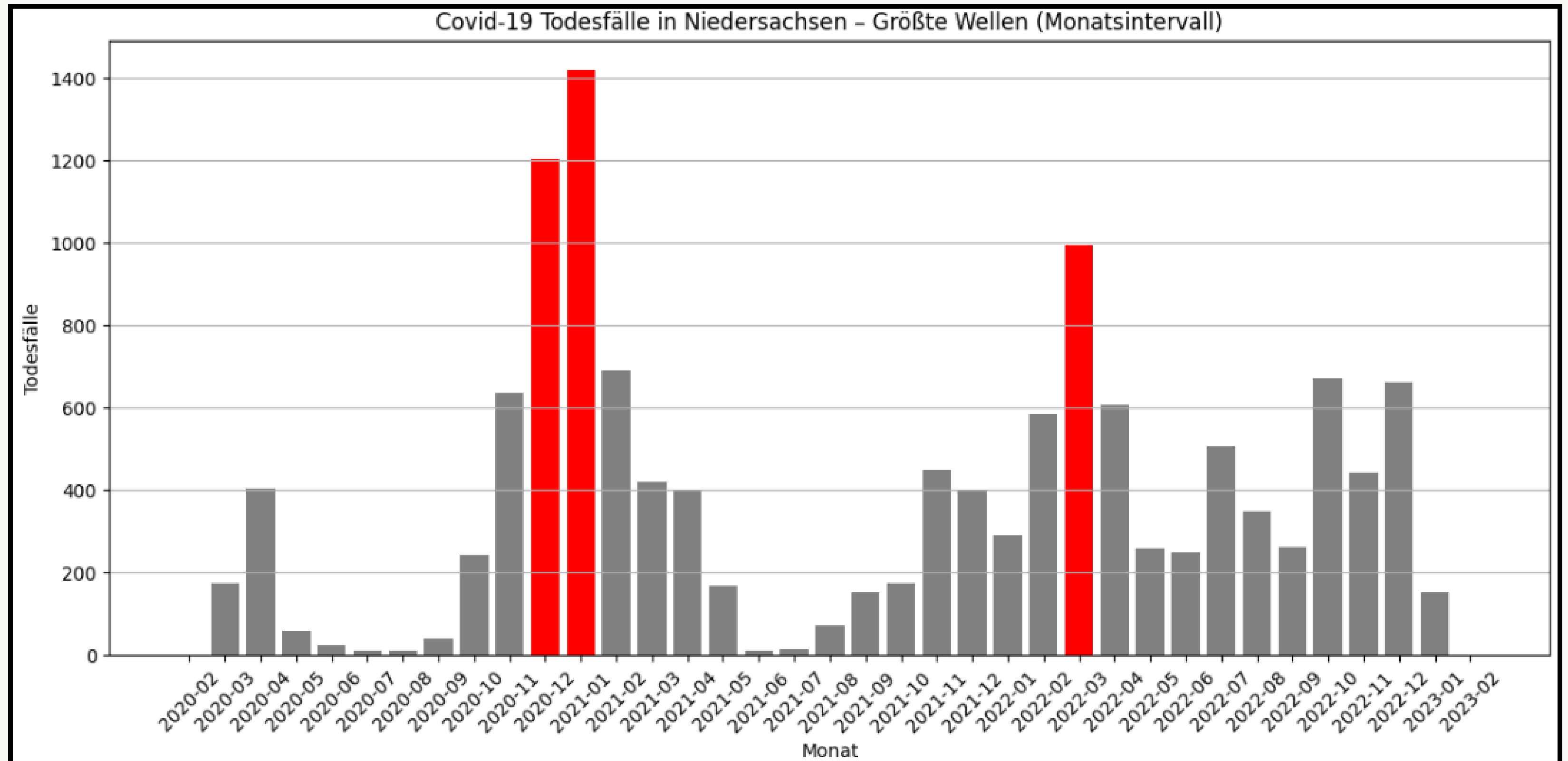
Die Plots zeigen:

- Mehrere COVID - Wellen in Niedersachsen
- Höhepunkt der Sterblichkeit im Winter 2020/21
- Deutlich sinkende relative Sterblichkeit im Zeitverlauf
- Vermutlich starker Einfluss durch Impfungen und Schutzmaßnahmen

AUFGABE 2B

Zeigen Sie das Verhältnis von Todesfällen zu Infektionen in Niedersachsen und ermitteln Sie den/ die Zeitraum /Zeiträume mit den meisten Todesfällen im Zusammenhang mit Covid-19.

ERGEBNIS



CODE

```
#Daten nach chronologischer Reihenfolge sortieren (Wiederholung, vermutlich nicht nötig)
df_nd = df_nd.sort_values(by='date')

#Monate aus Datum erzeugen, erstellen einer neuen Spalte "year_month": Datum in Monatsperiode (YYYY-MM)
df_nd['year_month'] = df_nd['date'].dt.to_period('M')

#Todesfälle pro Monat berechnen, Tage eines Monats zusammenfassen und aufsummieren
monthly_deaths = df_nd.groupby('year_month')['deaths'].sum().reset_index()

#3 Monate mit den meisten Todesfällen finden
top_3_months = monthly_deaths.nlargest(3, 'deaths')['year_month'].tolist()

#Farben für das Diagramm festlegen: Wenn Top 3 dann "rot", sonst "grau"
colors = ['red' if ym in top_3_months else 'grey' for ym in monthly_deaths['year_month']]

#Grafik erstellen
plt.figure(figsize=(14,6))
plt.bar(monthly_deaths['year_month'].astype(str), monthly_deaths['deaths'], color=colors)
plt.title('Covid-19 Todesfälle in Niedersachsen - Größte Wellen (Monatsintervall)')
plt.xlabel('Monat')
plt.ylabel('Todesfälle')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```

ZUSAMMENFASSUNG (2B)

Die Plots zeigen:

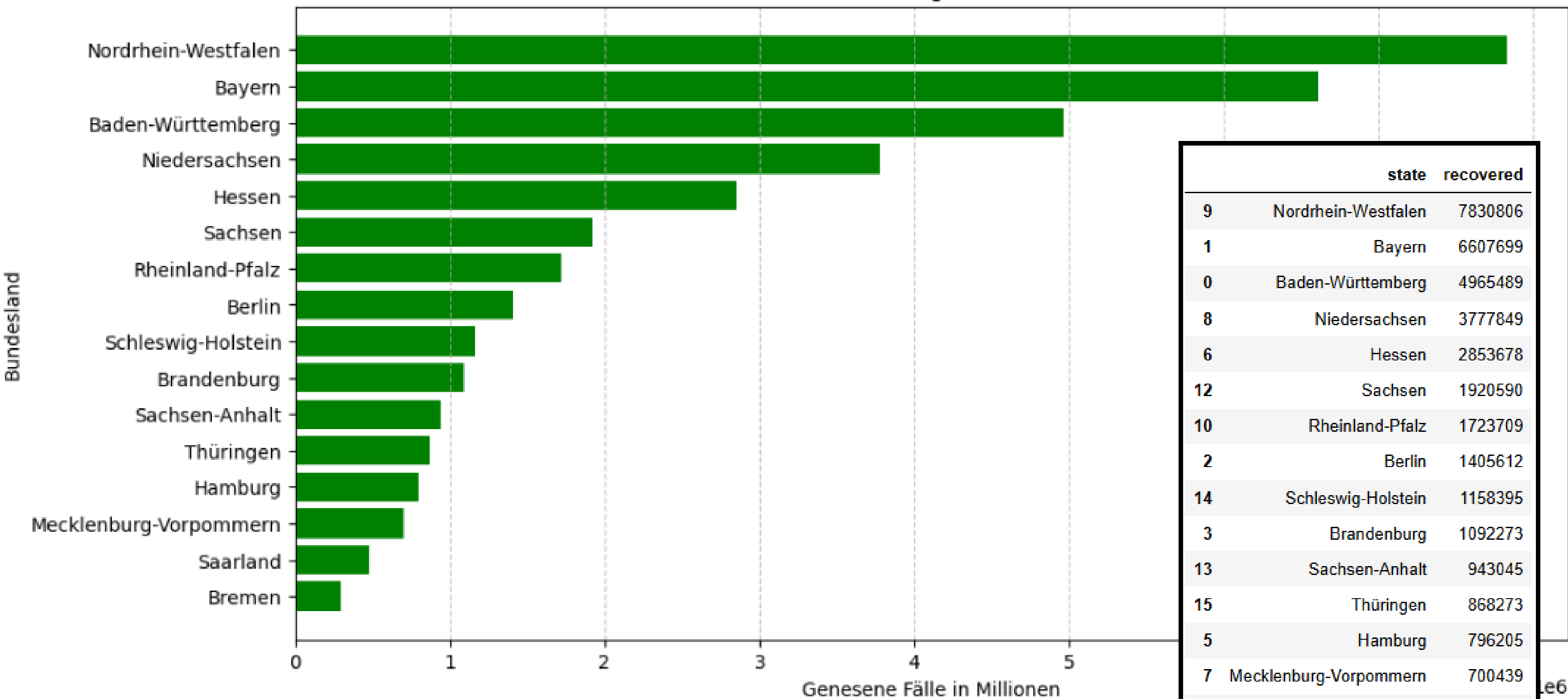
- COVID-19-Todesfälle in Niedersachsen, im Zeitraum 2020-2023 in Balkendiagramm
 - Rotmarkierte Balken zeigen drei Monate, mit den höchsten Todeszahlen
 - **Ergebnis:** Höchste Todeszahlen im Winter 2020/2021 & Beginn 2022
- COVID-19 verläuft in Wellen, wobei die stärkste Belastung im Winter erfolgt
- **Zusammengefasst bedeutet das:** Deutlich erkennbare Hochphasen der Todesrate in den Wintermonaten 2020/21 sowie Anfang 2022, was wohlmöglich auf saisonale Effekte zurückzuführen ist

AUFGABE 3

**Sortiere die Bundesländer nach ihren
Genesungszahlen in einer sinnvollen Reihenfolge.**

ERGEBNIS

Gesamtzahl der Covid-19 Genesungen nach Bundesland (in Millionen)



	state	recovered
9	Nordrhein-Westfalen	7830806
1	Bayern	6607699
0	Baden-Württemberg	4965489
8	Niedersachsen	3777849
6	Hessen	2853678
12	Sachsen	1920590
10	Rheinland-Pfalz	1723709
2	Berlin	1405612
14	Schleswig-Holstein	1158395
3	Brandenburg	1092273
13	Sachsen-Anhalt	943045
15	Thüringen	868273
5	Hamburg	796205
7	Mecklenburg-Vorpommern	700439
11	Saarland	480656
4	Bremen	298768

e6

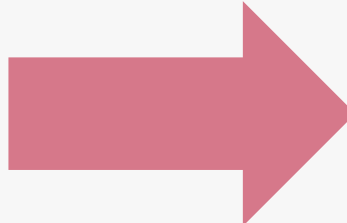
CODE

```
#CSV-Datei einlesen
df_state = pd.read_csv("covid_per_state.csv")

#Gesamt-Genesungen pro Bundesland berechnen
recovered_sum = df_state.groupby('state')['recovered'].sum().reset_index()

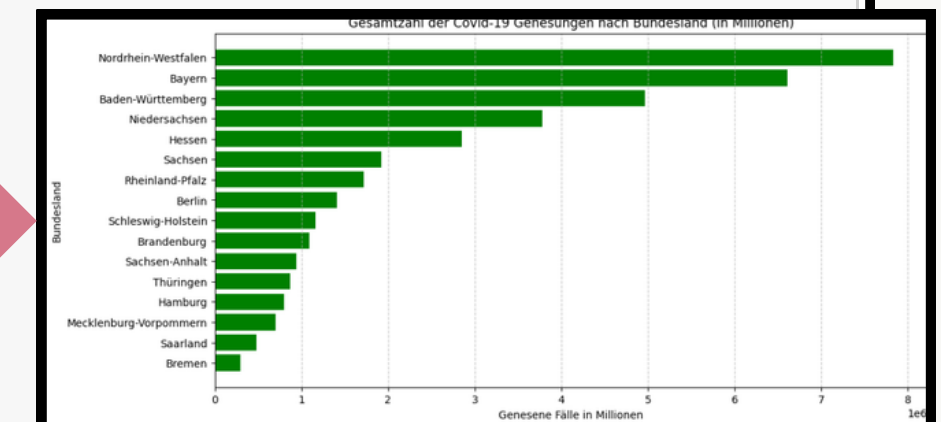
#Nach Genesungen absteigend sortieren
recovered_sorted = recovered_sum.sort_values(by='recovered', ascending=False)

#Ergebnis anzeigen
recovered_sorted
```



	state	recovered
9	Nordrhein-Westfalen	7830806
1	Bayern	6607699
0	Baden-Württemberg	4965489
8	Niedersachsen	3777849
6	Hessen	2853678
12	Sachsen	1920590
10	Rheinland-Pfalz	1723709
2	Berlin	1405612
14	Schleswig-Holstein	1158395
3	Brandenburg	1092273
13	Sachsen-Anhalt	943045
15	Thüringen	868273
5	Hamburg	796205
7	Mecklenburg-Vorpommern	700439
11	Saarland	480656
4	Bremen	298768

```
#Grafik zur Veranschaulichung aus den Daten erstellen
plt.figure(figsize=(12,6))
plt.barh(recovered_sorted['state'], recovered_sorted['recovered'], color='green')
plt.xlabel('Genesene Fälle in Millionen')
plt.ylabel('Bundesland')
plt.title('Gesamtzahl der Covid-19 Genesungen nach Bundesland (in Millionen)')
plt.gca().invert_yaxis()
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



ZUSAMMENFASSUNG

1.Code Abschnitt

- Gesamt-Genesung pro Bundesland
- absteigend sortiert
- Tabelle

2.Code Abschnitt

- horizontales Balkendiagramm
- Bremen: geringste Genesungen
- NRW: höchste Genesung

VIELEN

DANK