Projekt: Analysing Covid-19 Data

Aurela Brahimi, Izabela Mike, Lara Troschke und Inga-Brit Turschner

Übersicht

01. **Analyse des Datensets**

Datencheck, Hypothesen-Check und erste Impfwelle

02. **Mapping**

Folium-Mappe und Ranking der Bundesländer

03. **Fallstudie**

Lokales Minimum/Maximum, Korrelation,
Wahrscheinlichkeitsverteilung Genesung und Todesfall

01

ANALYSE DER DATENSETS

Datencheck

Hypothesen-Check

Erste Impfwelle

Datencheck

Was beinhalten die gegebenen Datensätze?

covid_de:	covid_per_state:	bundesländer.json:
pro Tag gemeldete Fälle, Genesene und Tote nach Bundesland (im Zeitraum Januar 2020 - Januar 2023)	Insgesamt gemeldete Fälle, Genesene und Tote nach Bundesland	GeoJSON-Daten mit Koordinaten zur Darstellung der deutschen Bundesländer als Multipolygone, inklusive Namen und ID der Bundesländer

Datencheck

Worauf kann geschlossen werden:

- Entwicklung der Fallzahlen (im zeitlichen Verlauf)
- Identifikation von Höhepunkten und Wellen
- Vergleiche zwischen Bundesländern
- Genesungs- und Todesrate
- Vergleich der Zahlen mit geografischer Lage

Worauf kann nicht sicher geschlossen werden:

- Auswirkungen des Virus
- Gründe für Verbreitung
- Impfkampagnen
- Corona Maßnahmen/Regelungen

→ Kausalzusammenhänge zwischen Fallzahlen und z.B. Lockdowns

Hypothese

„Covid-19-Impfungen haben keinen wesentlichen Beitrag zur Abschwächung der Pandemie geleistet!“

Wichtige Elemente im Code

Festlegen der Zeiträume, Gruppieren und berechnen der Daten:

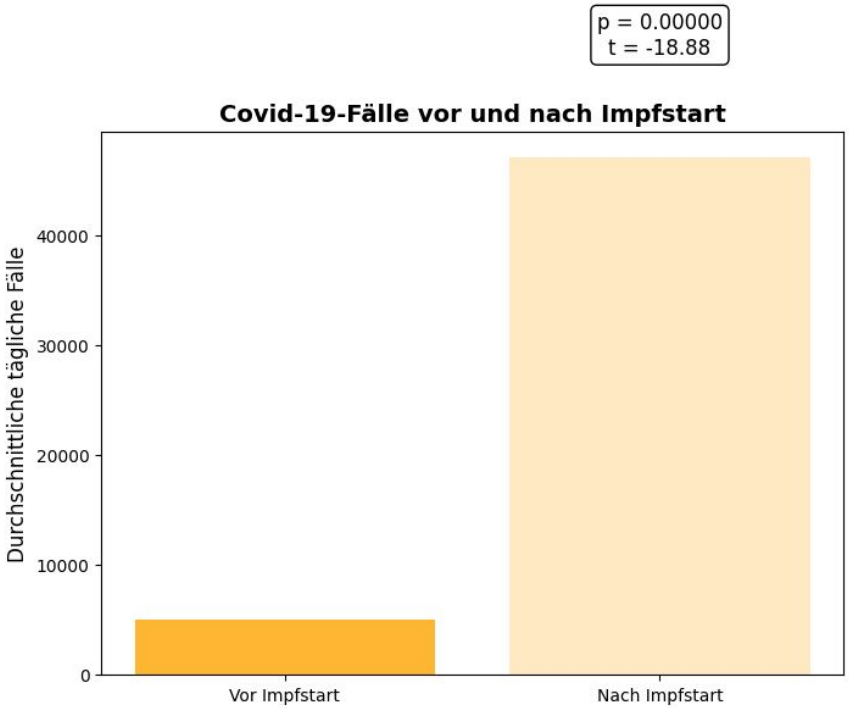
```
cutoff_date = pd.to_datetime("2020-12-27")
df_before = df[df['date'] < cutoff_date]
df_after = df[df['date'] >= cutoff_date]


daily_cases_before = df_before.groupby('date')['cases'].sum()
daily_cases_after = df_after.groupby('date')['cases'].sum()

daily_deaths_before = df_before.groupby('date')['deaths'].sum()
daily_deaths_after = df_after.groupby('date')['deaths'].sum()
```

Durchschnittswerte berechnen:

```
mean_cases_before = daily_cases_before.mean()
mean_cases_after = daily_cases_after.mean()

mean_deaths_before = daily_deaths_before.mean()
mean_deaths_after = daily_deaths_after.mean()
```

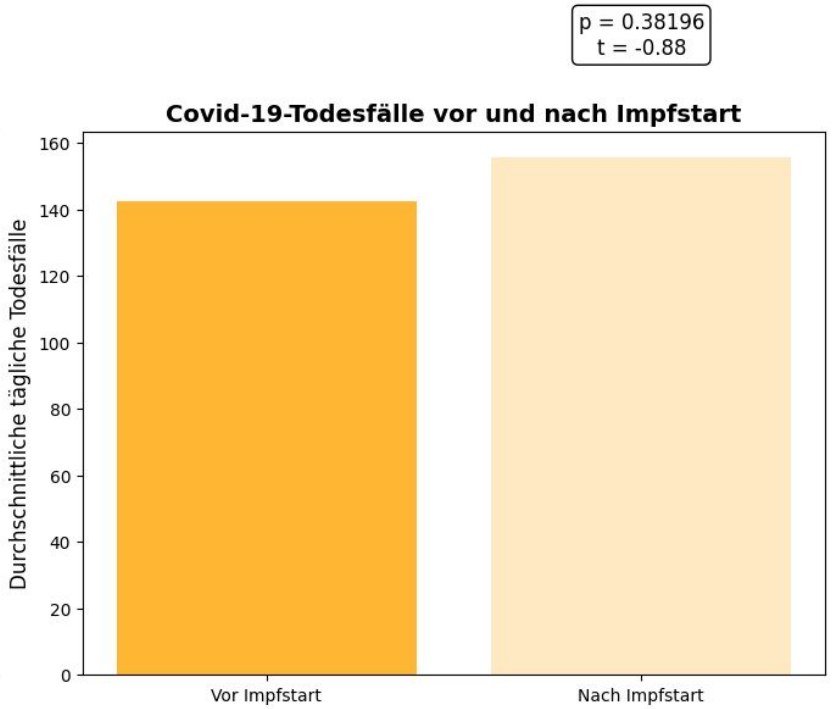
T-Test durchführen:

```
t_stat_cases, p_value_cases = stats.ttest_ind(daily_cases_before, daily_cases_after, equal_var=False)
t_stat_deaths, p_value_deaths = stats.ttest_ind(daily_deaths_before, daily_deaths_after, equal_var=False)
```



vor Impfstart: 5.062 Fälle pro Tag

nach Impfstart: 47.082 Fälle pro Tag



vor Impfstart: 142 Todesfälle pro Tag

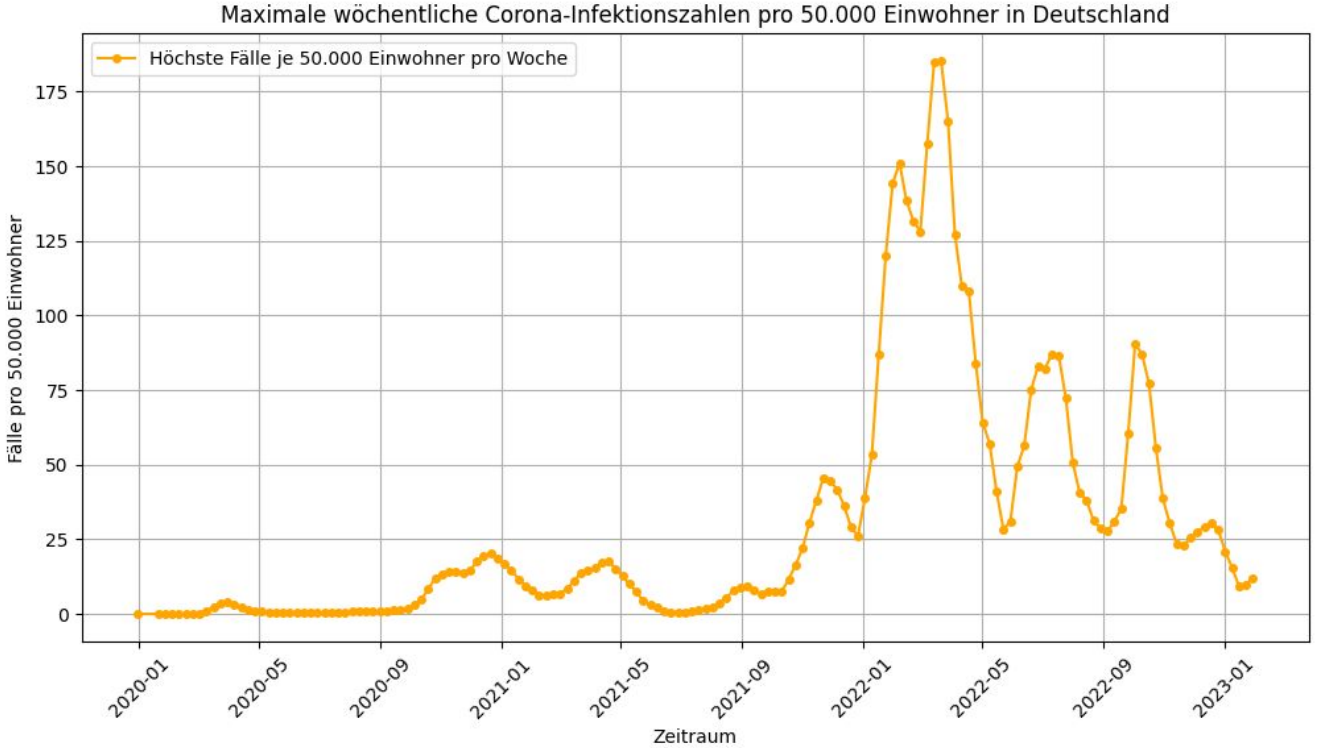
nach Impfstart: 156 Todesfälle pro Tag

Wichtige Elemente im Code

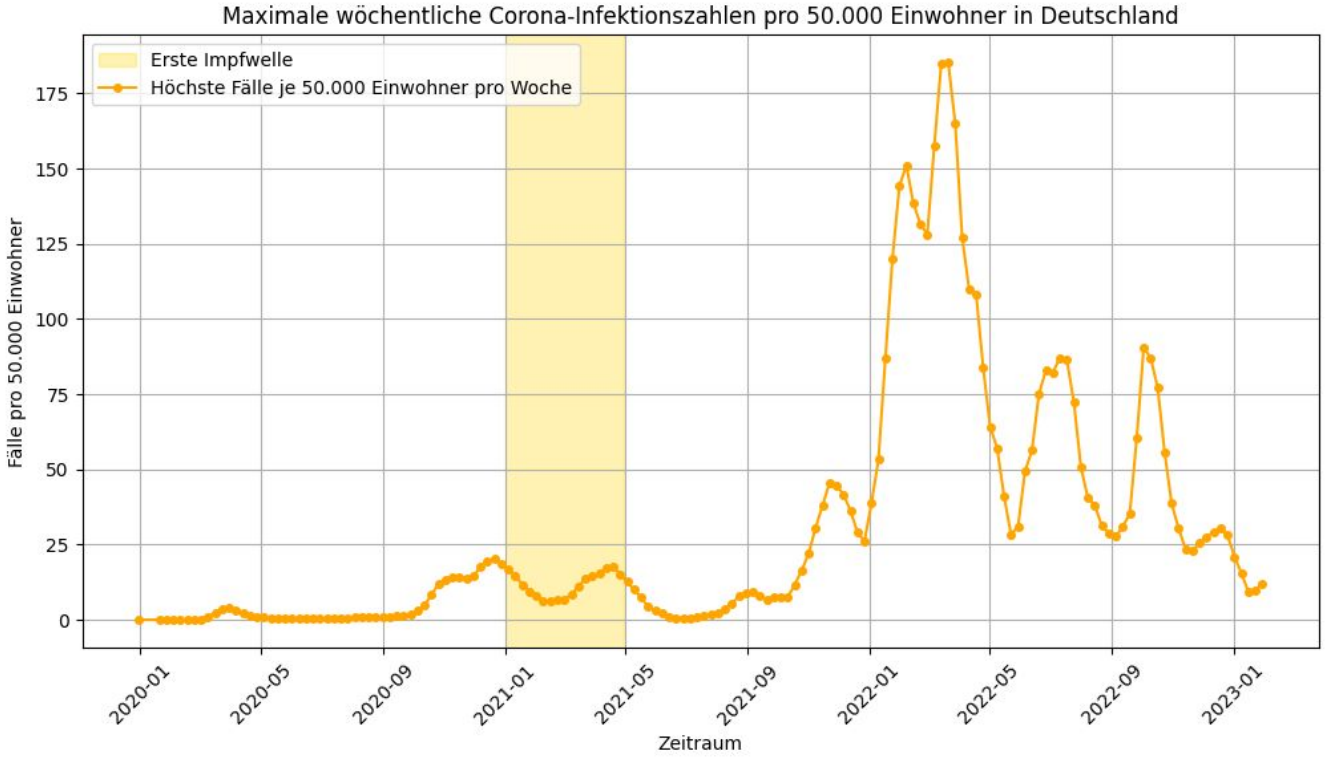
Fälle auf 50.000 Einwohner, Wochen & Maxima:

```
df_deutschland['cases_per_50000'] = df_deutschland['cases'] / (83100000 / 50000)

df_deutschland['week'] = df_deutschland['date'].dt.to_period('W')
df_max_weekly = df_deutschland.groupby('week')['cases_per_50000'].max().reset_index()

df_max_weekly['date'] = df_max_weekly['week'].dt.start_time
```

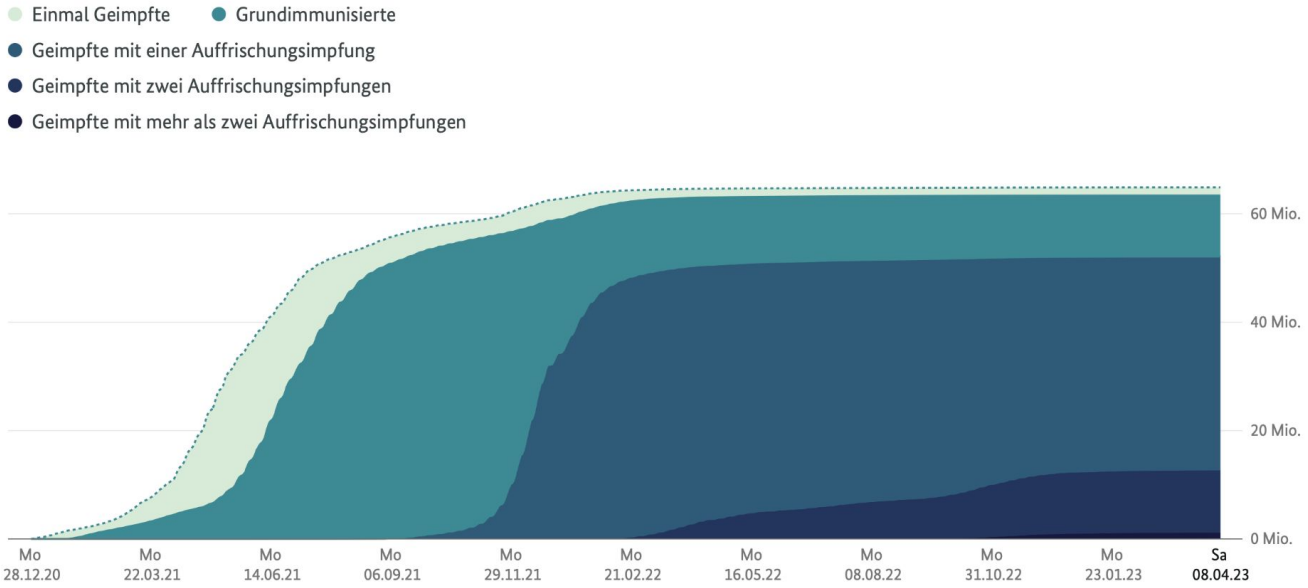
Infektionszahlenverlauf



Erste Impfwelle



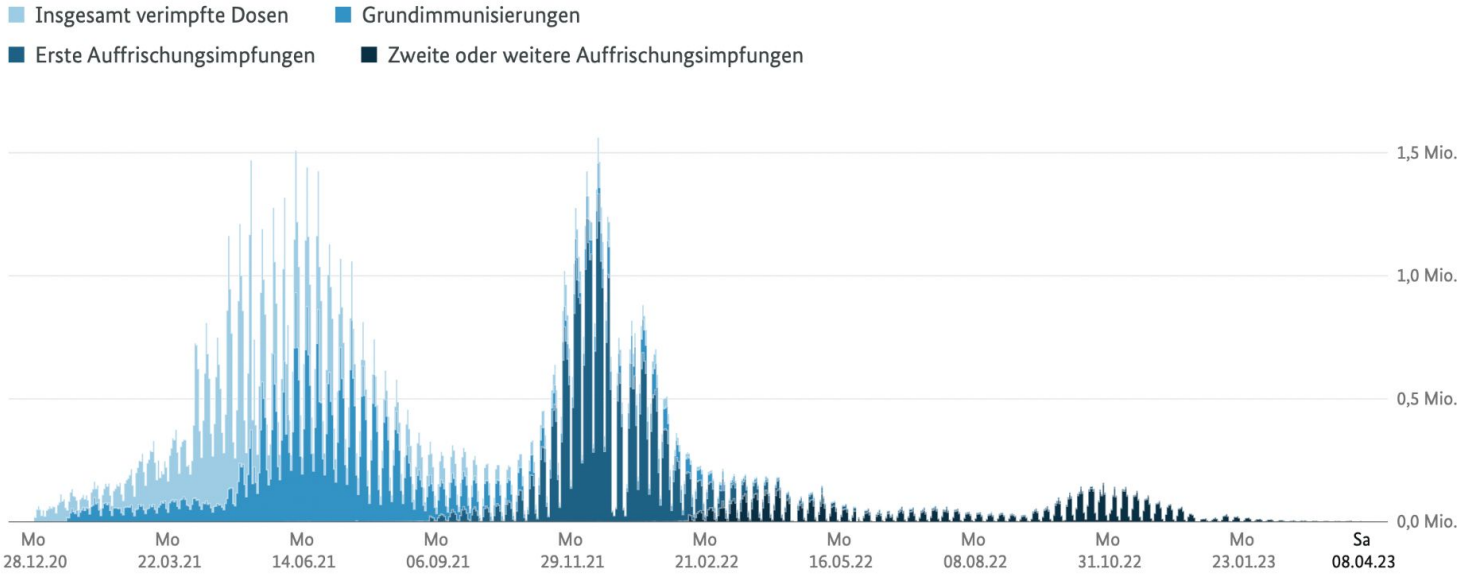
Erste Impfwelle - Vergleich mit offiziellen Daten



Stand: 8. April 2023 (Impfungen)
Quelle: impfdashboard.de, RKI, BMG.

Grafik 1

Erste Impfwelle - Vergleich mit offiziellen Daten



Stand: 8. April 2023 (Impfungen)
Quelle: impfdashboard.de, RKI, BMG.

Grafik 2

02

MAPPING

Folium-Mappe

Todesrate in Niedersachsen

Ranking der Bundesländer

Wichtige Elemente im Code

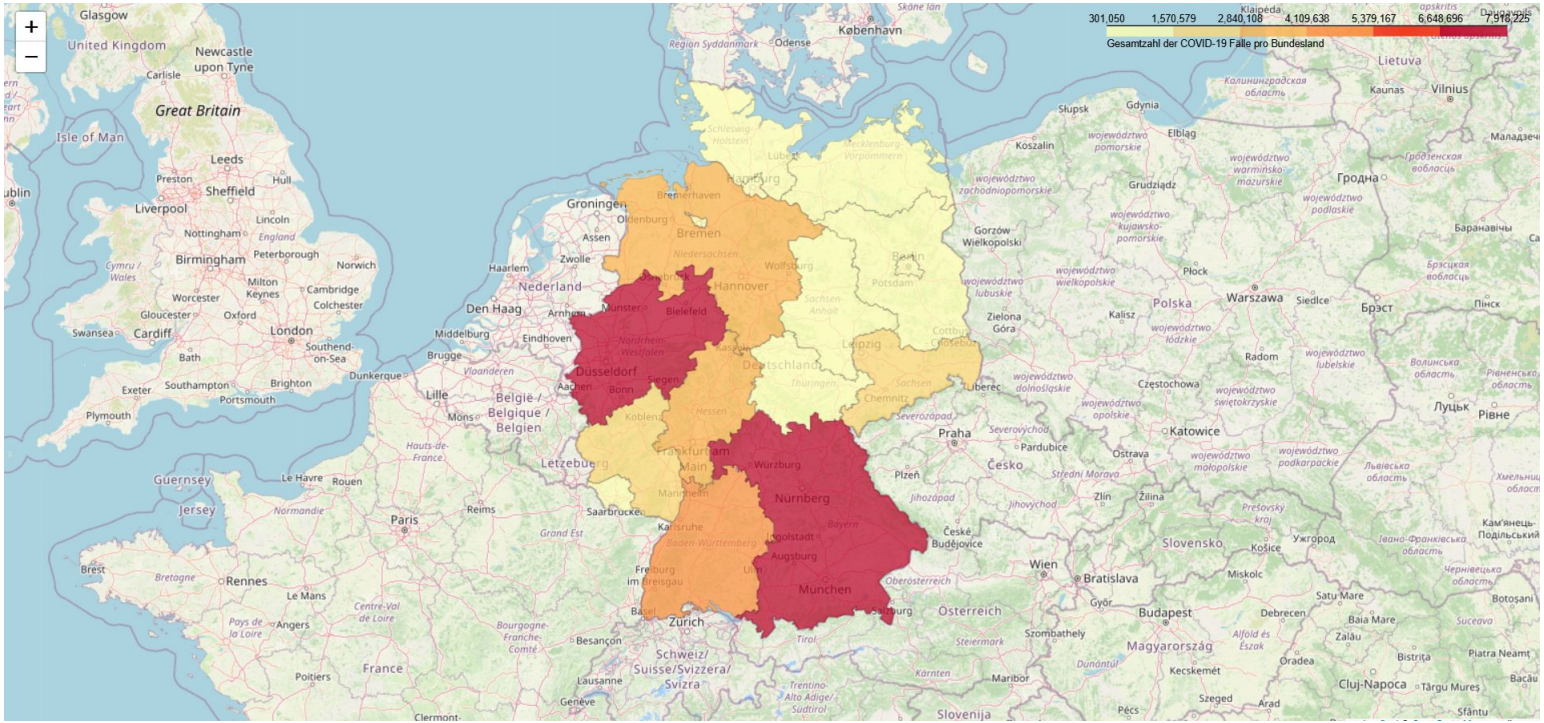
Verknüpfung der Covid-19 und der geografischen Daten:

```
for feature in geojson_data["features"]:
    state_name = feature["properties"]["name"]
    state_data = df_covid_state[df_covid_state["state"] == state_name]
    if not state_data.empty:
        cases = int(state_data["cases"])
        feature["properties"]["cases"] = cases
```

Map erstellen, Tooltips hinzufügen:

```
folium.Choropleth(
    geo_data=geojson_data,
    name="COVID-19 Fälle",
    data=df_covid_state,
    columns=["state", "cases"],
    key_on="feature.properties.name",
    fill_color="YlOrRd",
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name="Gesamtzahl der COVID-19 Fälle pro Bundesland"
).add_to(germany_map_cases)

folium.GeoJson(
    geojson_data,
    style_function=lambda feature: {
        "fillColor": "transparent",
        "color": "black",
        "weight": 0.5,
        "fillOpacity": 0,
    },
    tooltip=folium.GeoJsonTooltip(
        fields=["name", "cases"],
        aliases=["Bundesland: ", "Fälle: "],
        localize=True
    ),
).add_to(germany_map_cases)
```

Gesamtzahl der Covid-19 Fälle - Map



Todesfallrate in Niedersachsen

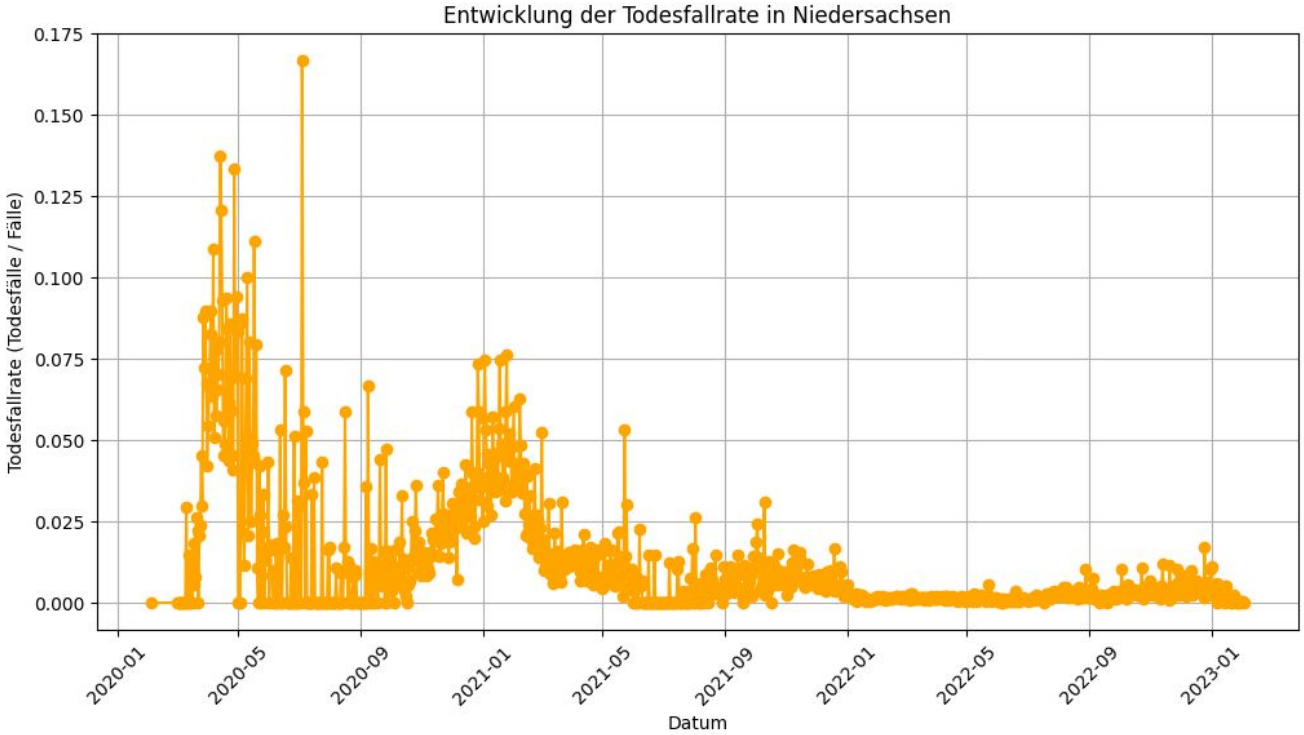
Todesfallrate = Todesfälle/Fälle

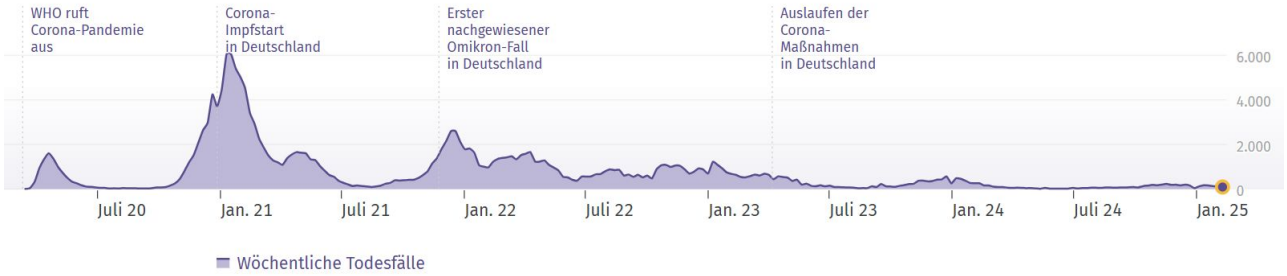
```
lower_saxony_data["death_to_case_ratio"] = lower_saxony_data["deaths"] / lower_saxony_data["cases"]
```

Visualisierung der höchsten Todesfallraten

```
highest_death_rate = lower_saxony_data.nlargest(5, "death_to_case_ratio")
print(highest_death_rate)
```

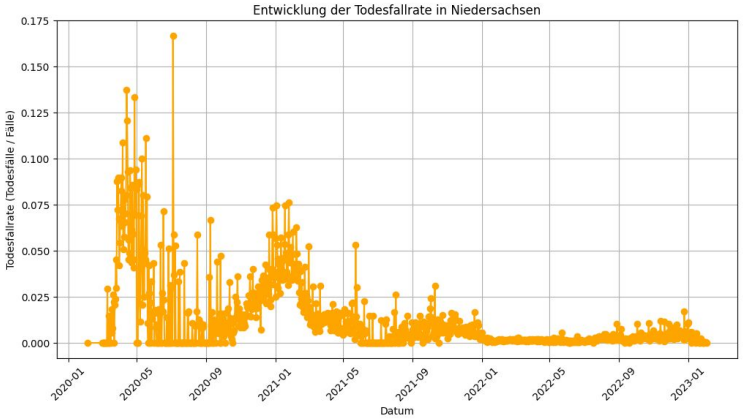
Todesfallrate in Niedersachsen





Wöchentliche Zahl der COVID-19-Todesfälle. Aktualisiert am 15.02.2025. Das nächste Update wird zum 22.02.2025 erwartet.

Quelle: Bundesministeriums für Gesundheit, Robert Koch-Institut
Daten: RKI

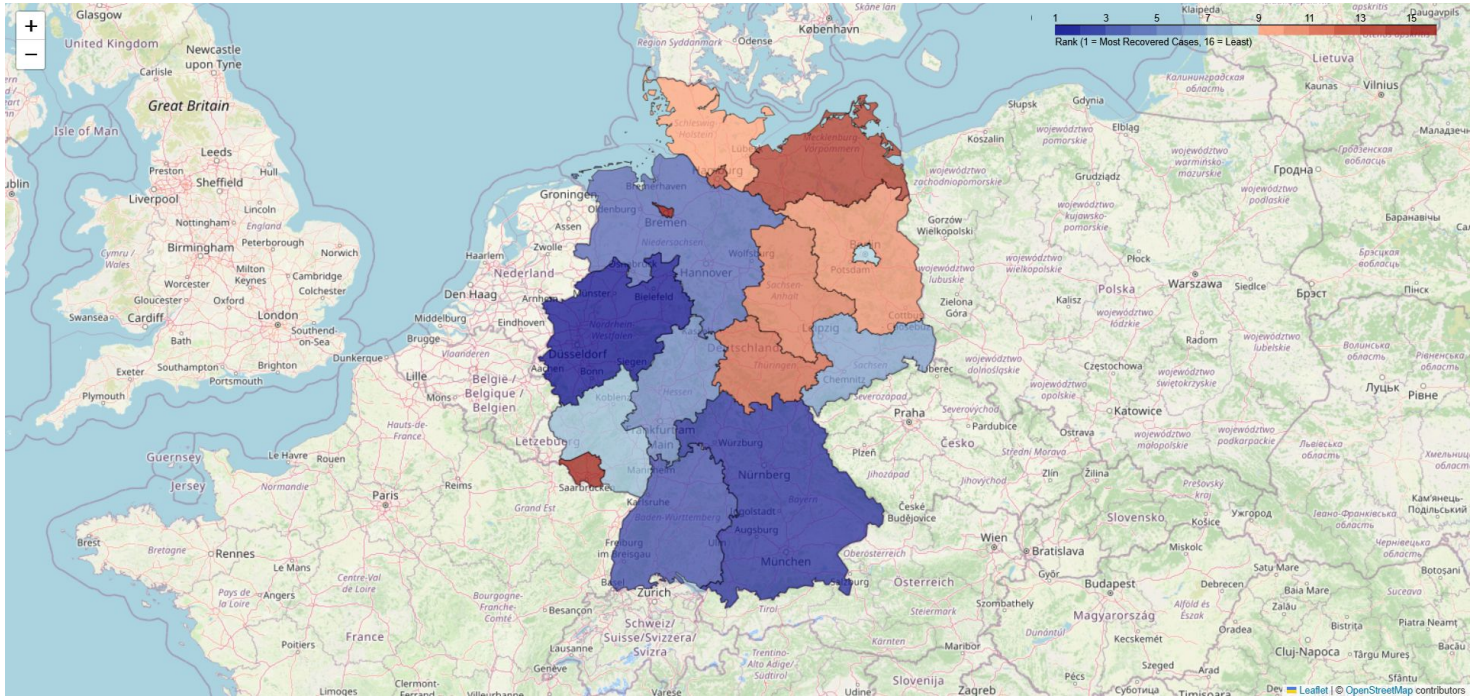


Rang der Bundesländer nach Genesenenanzahl

Erstellen einer Rangordnung & Verknüpfung der Covid-19 Daten mit den geografischen Daten

```
df_covid_state = pd.read_csv('covid_per_state.csv')

state_df = df_covid_state.groupby('state').agg(
    recovered=('recovered', 'sum')
).reset_index()

state_df['recovered_rank'] = state_df['recovered'].rank(method='dense', ascending=False)

for feature in geojson_data["features"]:
    state_name = feature["properties"]["name"]
    rank = int(state_df.loc[state_df['state'] == state_name, 'recovered_rank'].values[0])
    recovered = int(state_df.loc[state_df['state'] == state_name, 'recovered'].values[0])
    feature["properties"]["rank"] = rank
    feature["properties"]["recovered"] = recovered
```

Rang der Bundesländer nach Genesenenanzahl



03

FALLSTUDIE

Lokales Minimum/Maximum

Korrelation

Wahrscheinlichkeitsverteilung Genesung und Todesfälle

Wichtige Elemente im Code

Daten für ganz Deutschland aggregieren

```
df_germany = df.groupby("date")[["cases", "recovered", "deaths"]].sum().reset_index()
```

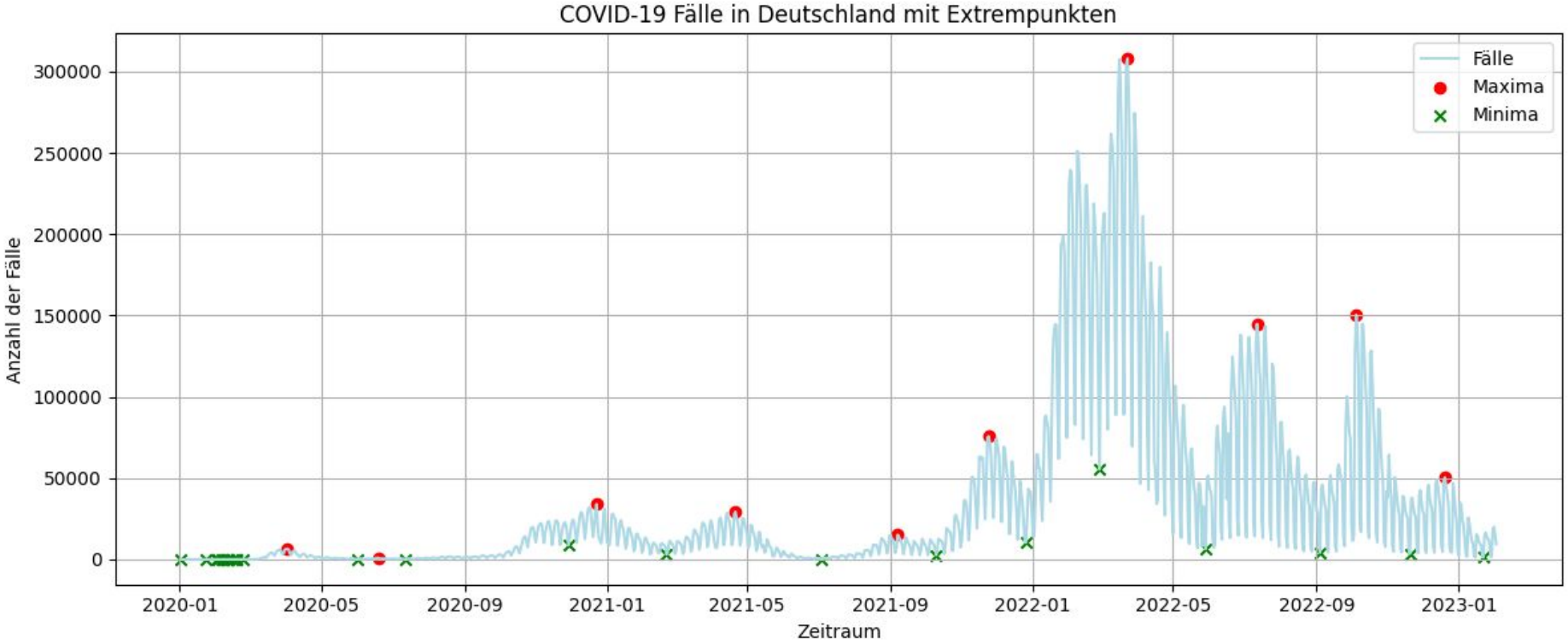
Funktion zur Ermittlung lokaler Extrempunkte:

```
def find_extrema(data, order=30):
    """
    Findet lokale Maxima und Minima in einer Zeitreihe mit einer bestimmten Glättung (order)
    """
    maxima_idx = argrelextrema(data.values, np.greater_equal, order=order)[0]
    minima_idx = argrelextrema(data.values, np.less_equal, order=order)[0]
    return data.iloc[maxima_idx], data.iloc[minima_idx]
```

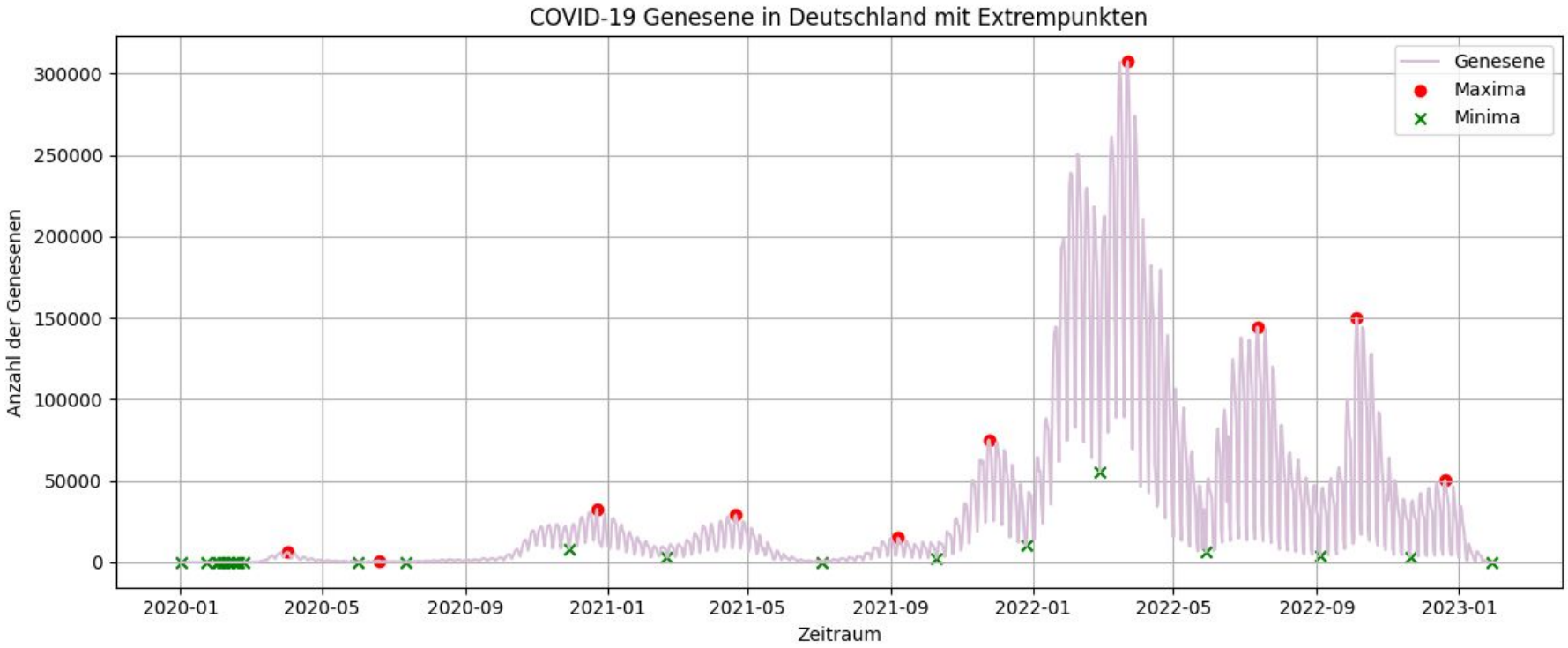
Bestimmen der Extrempunkte für verschiedene Variablen:

```
cases_max, cases_min = find_extrema(df_germany["cases"])
recovered_max, recovered_min = find_extrema(df_germany["recovered"])
deaths_max, deaths_min = find_extrema(df_germany["deaths"])
```

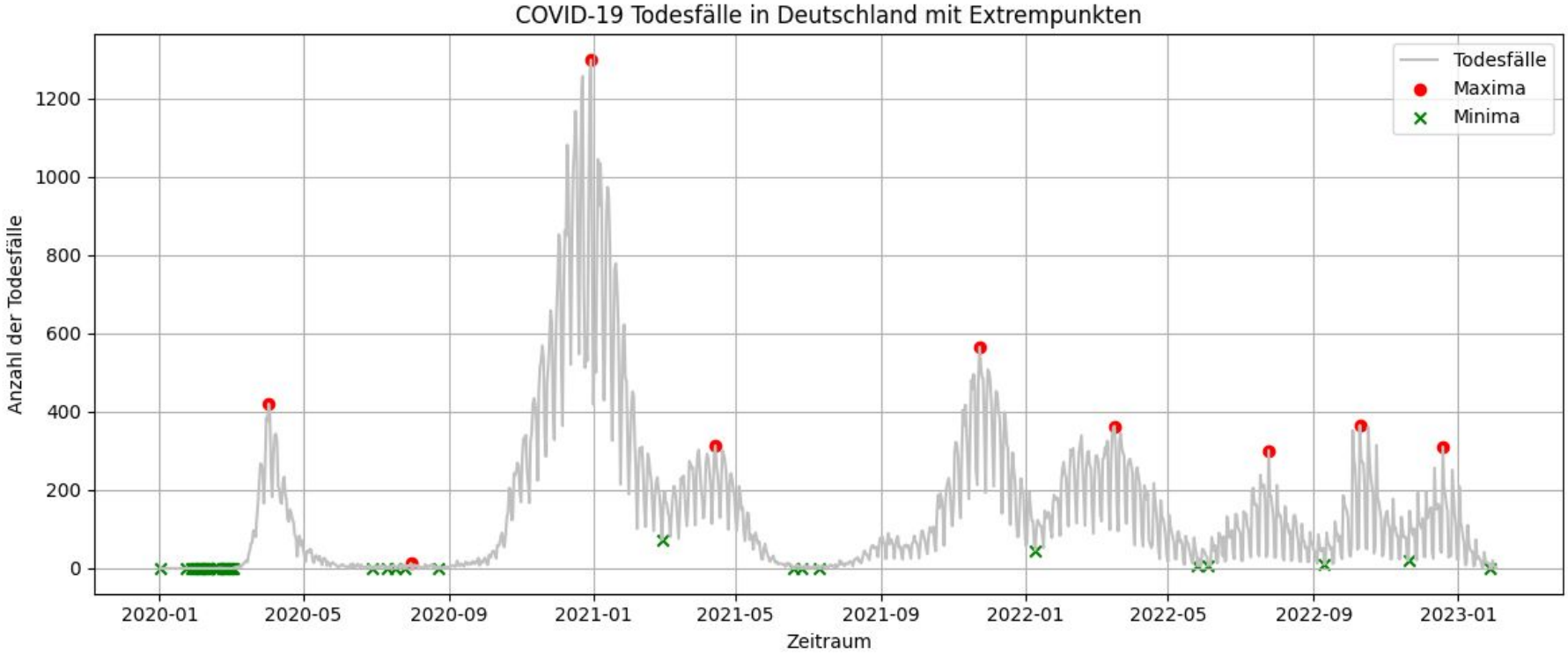
Lokale Extrempunkte der Covid-19 Fällen

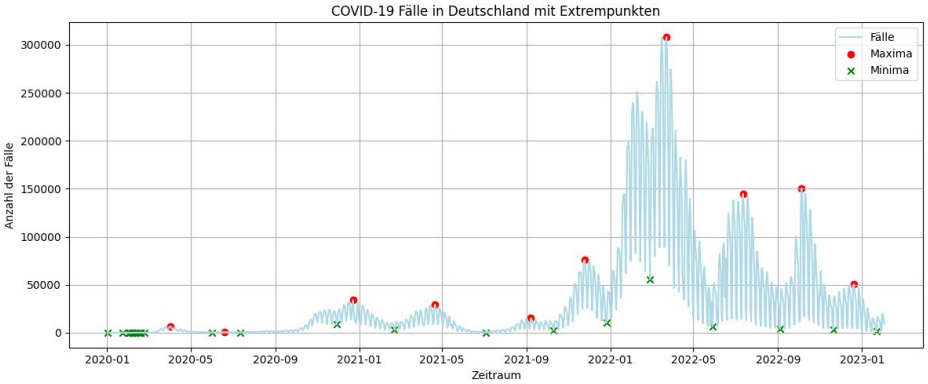


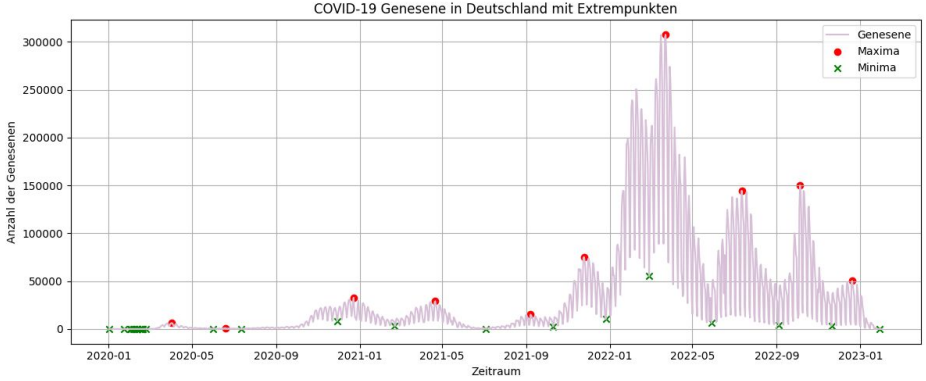
Lokale Extrempunkte der Genesenen Fälle

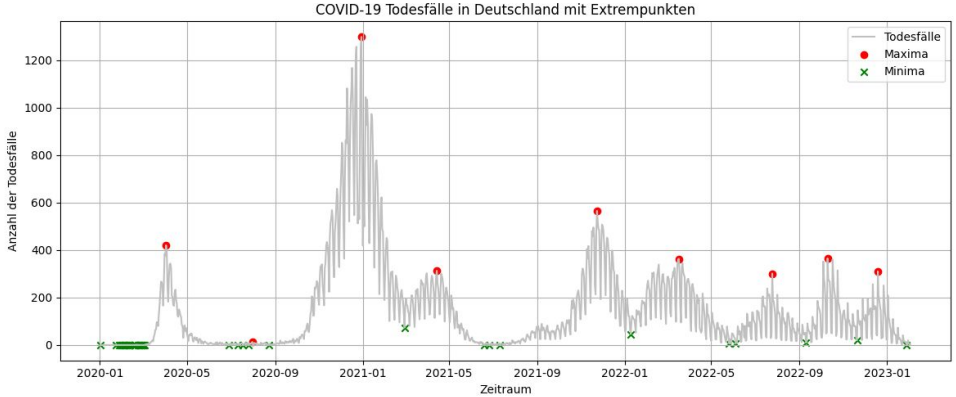


Lokale Extrempunkte der Todesfälle




COVID-19 Fälle in Deutschland mit Extrempunkten
Fälle
Maxima
Minima
Anzahl der Fälle
Zeitraum
0
50000
100000
150000
200000
250000
300000
2020-01
2020-05
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01


COVID-19 Genesene in Deutschland mit Extrempunkten
Genesene
Maxima
Minima
Anzahl der Genesenen
Zeitraum
0
50000
100000
150000
200000
250000
300000
2020-01
2020-05
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01


COVID-19 Todesfälle in Deutschland mit Extrempunkten
Todesfälle
Maxima
Minima
Anzahl der Todesfälle
Zeitraum
0
200
400
600
800
1000
1200
2020-01
2020-05
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01

Wichtige Elemente im Code

Spearman-Test Korrelation (E3.2):

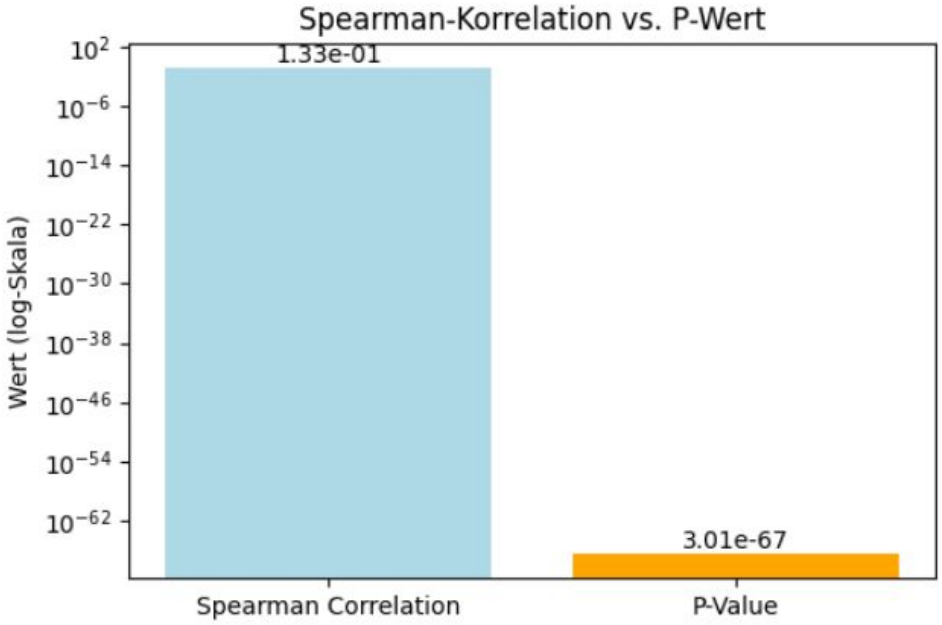
```
import pandas as pd
from scipy.stats import spearmanr


file_path = "/files/einfhrung-in-die-prog/Project-Geomapper/covid_de.csv"

df["date"] = pd.to_datetime(df["date"])

covid_data = df.sort_values(by="date")

covid_data["date_numeric"] = covid_data["date"].view("int64")

corr, p_value = spearmanr(covid_data["date_numeric"], covid_data["deaths"])

print(f"Spearman correlation: {corr}, p-value: {p_value}")
```

Wichtige Elemente im Code

Gesamtwerte und Wahrscheinlichkeiten berechnen (E3.3):

```
# Gesamtwerte berechnen für den neuen Datensatz
total_cases_new = df_new["cases"].sum()
total_deaths_new = df_new["deaths"].sum()
total_recovered_new = df_new["recovered"].sum()

# Wahrscheinlichkeiten berechnen
death_rate = total_deaths / total_cases
recovery_rate = total_recovered / total_cases

# Ergebnisse ausgeben
total_cases, total_deaths, total_recovered, death_rate, recovery_rate
```

Korrelation Datum und Todesrate



Korrelation = 0.1327, p = 3.0064e-67

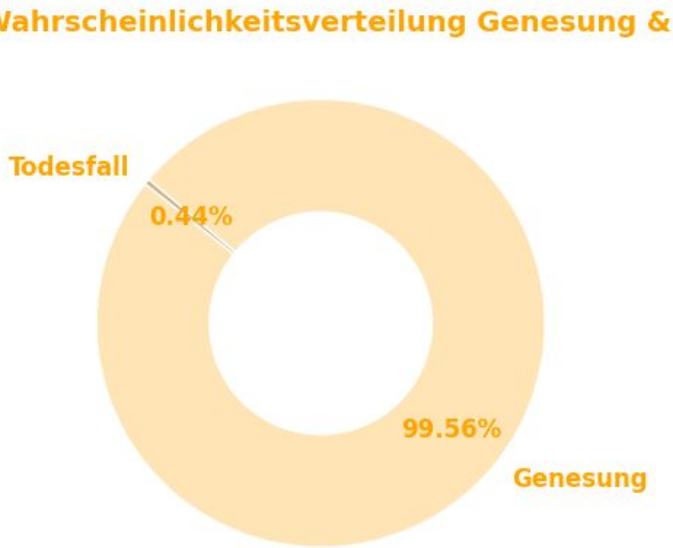
- Schwache positive Korrelation zwischen Datum und Todeszahlen.
- Sehr kleiner p-Wert → Die Korrelation ist statistisch hochsignifikant, also kein Zufall.

Wie wahrscheinlich ist eine Genesung oder ein Todesfall?

37.81 Mio
infizierte Personen

37.42 Mio
genesene Personen

166.016 Tsd
verstorbene Personen



Quellenverzeichnis

https://impfdashboard.de/#:~:text=April%202023-,64%2C9%20Mio.,(76%2C4%20%25)%20grundimmunisiert. [zuletzt aufgerufen am 07.02.2025]

https://www.bundesgesundheitsministerium.de/coronavirus/chronik-coronavirus.html [zuletzt aufgerufen am 12.02.2025]

https://infektionsradar.gesund.bund.de/de/covid/todesfaelle [zuletzt aufgerufen am 12.02.2025]

Grafik 1 & 2:

https://impfdashboard.de/ [zuletzt aufgerufen am 13.02.2025]

Dankeschön für Ihre Aufmerksamkeit!