Analysing Covid-19 Data:

Gruppenprojekt Medienwissenschaften

Janna Heiny, Milena Krieger, Leonie Manthey, Natascha Rott & Sarina Apel

E1 - Analyzing the Data

What is the Data about?

- covid_de.csv: enthält Daten zu Covid-19 Fällen In Deutschland –> Bundesländer von Deutschland, Erkrankungsdatum, Anzahl der Infizierten, Genesenen und Todesfälle
- covid_per_state.csv: zusammengefasste Daten zu Covid-19-Fällen auf Bundeslandebene –> Gesamtzahlen der Erkrankten, Genesenen und Verstorbenen pro Bundesland.
- bundeslaender.json: Koordinaten der Bundesländer(-grenzen)

E1 - Analyzing the Data

What should and not should be concluded from the Data?

- Should concluded: Verlauf der Pandemie –> regionale und bundesweite Unterschiede, Entwicklung der Infizierungen, Genesungen und Todesfällen im zeitlichen Verlauf

- Should not concluded: Kausale Zusammenhänge, Dunkelziffern und individuelle Informationen (Infektionsrisiko, Symptome die bei den Covid-19 erkrankten aufgetreten sind und woran die Personen dann gestorben sind, wie Alt die Personen waren, ob sie geimpft waren oder nicht)

E1 - Analyzing the Data

Hypotheses: “Covid-19 vaccinations have not made a significant contribution to the easing of the pandemic!"

➡ Erste Impfung war am 27.12.2020: Um zu schauen ob es unmittelbar nach der ersten Impfung unterschiede gab –> Zeitraum 02.01.2020 - 26.12.2020 und 27.12.2020 - 02.02.2023

```
# Durchschnittliche Covid-Fälle für die Zeiträume vor und nach der ersten Impfung
durchschnitt_vor_impfung = covid_vor_impfung["cases"].mean()
durchschnitt_nach_impfung = covid_nach_impfung["cases"].mean()
```

```
# Gesamtfälle in den jeweiligen Zeiträumen
gesamtfälle_vor_impfung = covid_vor_impfung["cases"].sum()
gesamtfälle_nach_impfung = covid_nach_impfung["cases"].sum()

# Anteile in Relation berechnen
relativer_mittelwert_vor = (durchschnitt_vor_impfung / gesamtfälle_vor_impfung) * 100
relativer_mittelwert_nach = (durchschnitt_nach_impfung / gesamtfälle_nach_impfung) * 100
```

Ausgabe:

```
Mittelwert für die Erkrankung vor der ersten Impfung: 351.43
Mittelwert für die Erkrankung nach der ersten Impfung: 2961.93

Covid-19 Fälle im Verhältnis zur Gesamtanzahl vor der ersten Impfung (relativer Mittelwert): 0.0213%
Covid-19 Fälle im Verhältnis zur Gesamtanzahl nach der ersten Impfung (relativer Mittelwert): 0.0082%
```

E1 - Analyzing the Data

Hypotheses: “Covid-19 vaccinations have not made a significant contribution to the easing of the pandemic!"

Covid-19-Fälle vor und nach der ersten Impfung (Zeitraum: 02.01.2020 - 26.12.2020 und 27.12.2020 - 02.02.2023)

```
# Berechnung der "tage_seit_start" für beide Zeiträume (nummerische Werte)
tage_seit_start_vor_impfung = (covid_vor_impfung['date'] - covid_vor_impfung['date'].min()).dt.days
tage_seit_start_nach_impfung = (covid_nach_impfung['date'] - covid_nach_impfung['date'].min()).dt.days

# Berechnung der Spearman-Korrelation für den Zeitraum vor der Impfung
spearman_corr_vor, p_value_vor = stats.spearmanr(tage_seit_start_vor_impfung, covid_vor_impfung['cases'])

print(f"\nSpearman-Korrelation vor der ersten Impfung: {spearman_corr_vor}")
print(f"p-Wert vor der ersten Impfung: {p_value_vor}")

# Berechnung der Spearman-Korrelation für den Zeitraum nach der Impfung
spearman_corr_nach, p_value_nach = stats.spearmanr(tage_seit_start_nach_impfung, covid_nach_impfung['cases'])

print(f"\nSpearman-Korrelation nach der ersten Impfung: {spearman_corr_nach}")
print(f"p-Wert nach der ersten Impfung: {p_value_nach}")
```

Beispiel: erstes Datum 02.01.2020 & weiteres Datum 05.01.2020 –> Unterschied 3 Tage

1. Zeitraum: vor der Impfung

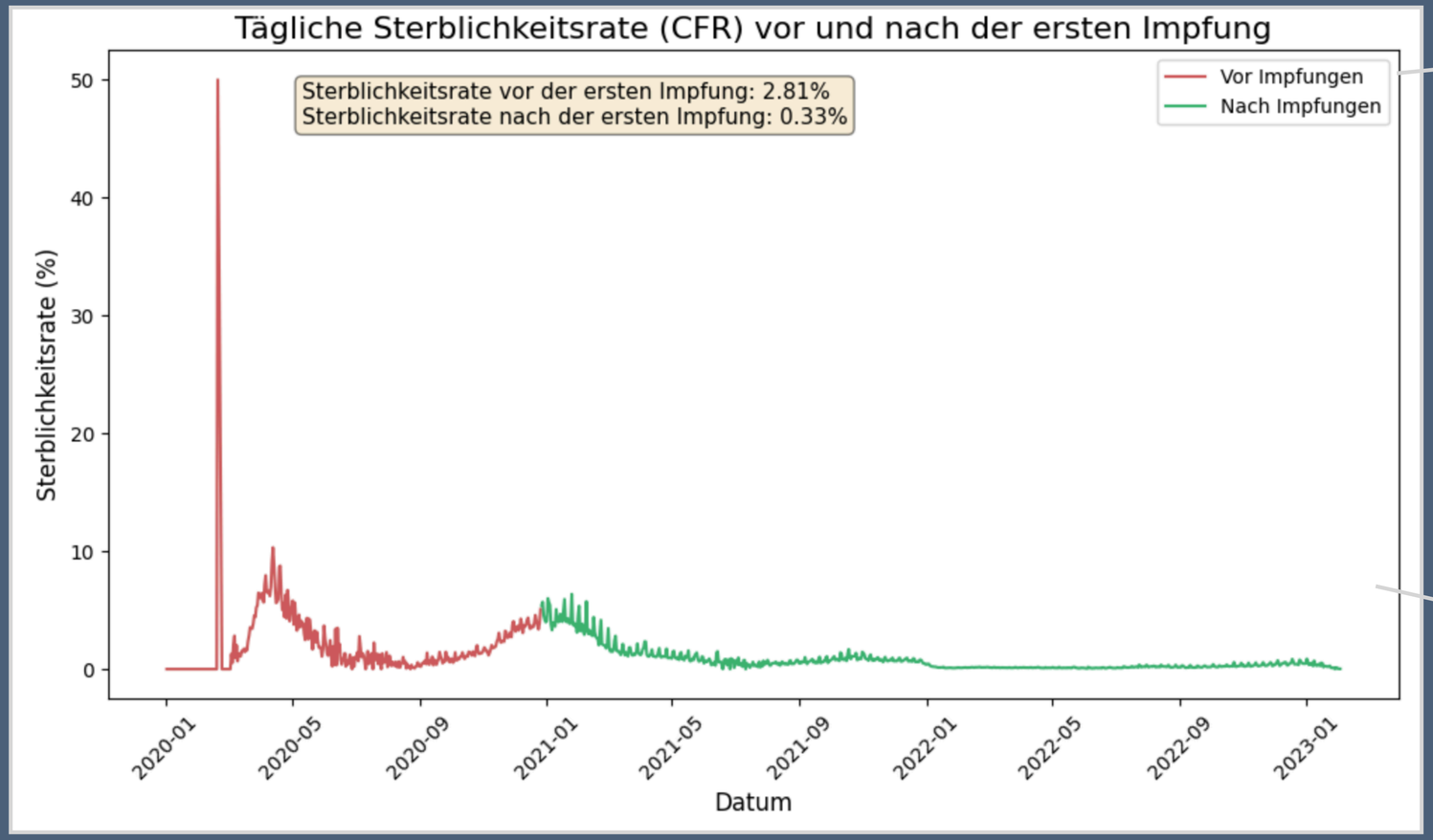
2. Zeitraum: nach der Impfung

Ausgabe:

```
Spearman-Korrelation vor der ersten Impfung: 0.4961, p-Wert vor der ersten Impfung: 0.0000
Spearman-Korrelation nach der ersten Impfung: 0.3239, p-Wert nach der ersten Impfung: 0.0000
```

E1 - Analyzing the Data

Hypotheses: “Covid-19 vaccinations have not made a significant contribution to the easing of the pandemic!"



Sterblichkeitsrate maximum vor der ersten Impfung: ~50%

Sterblichkeitsrate maximum nach der ersten Impfung: < 10%

E1 - Analyzing the Data

the first vaccination wave from the given data and the periods given by official bodies

```python
# Schwellenwert für signifikanten Anstieg der Gesamtveränderung (95%-Quantil)
schwellenwert = covid_de['ausgleich_veränderung'].quantile(0.95)

# Zeitraum der ersten Impfwelle bestimmen (basierend auf den Daten)
impfstart_daten = covid_de[covid_de['ausgleich_veränderung'] > schwellenwert]['date']
if not impfstart_daten.empty:
    impfstart_beginn = impfstart_daten.min().date()
    impfstart_ende = impfstart_daten.max().date()
    print(f"Geschätzter Zeitraum der ersten Impfwelle: {impfstart_beginn} bis {impfstart_ende}")
else:
    impfstart_beginn = None
    impfstart_ende = None
    print("Kein signifikanter Zeitraum für die erste Impfwelle gefunden.")
```

Ausgabe:

Vergleich des geschätzten und offiziellen Impfstarts

```
Geschätzter Zeitraum der ersten Impfwelle: 2021-11-24 bis 2022-10-25
Offizielle erste Impfwelle laut RKI: 2021-01-01 bis 2021-04-30
```

E2 - Mapping

Data Used


```
{'type': 'Feature',
 'id': 4,
 'properties': {'id': 'DE-HB', 'name': 'Bremen', 'type': 'State'},
 'geometry': {'type': 'Polygon',
  'coordinates': [[[8.98544883728033, 53.128219604492244],
    [8.973159790039006, 53.12799072265631],
    [8.967188835144043, 53.12414932250988],
    [8.948739051818961, 53.12380218505865],
    [8.94909858703619, 53.11632919311535],
    [8.955239295959586, 53.11645126342779],
                                    ...
```


Ausschnitt aus der GeoJSON-Datei am Beispiel Bremen

Ziel: Einbindung der Daten zu Covid-19

Data Used

1. Erstellen einer Farbskala

```
colormap = linear.Reds_09.scale(
    covid_per_state.cases.min(), covid_per_state.cases.max()
)
```

2. Daten filtern und Erstellen eines Dictionaries

```
data_states_dict = covid_per_state.set_index("state")["cases"]
```

➡ Hier: Covid-19 Fälle je Bundesland

3. Hervorhebungen erstellen

```
popup = folium.GeoJsonPopup(
    fields=["name","cases", "deaths", "recovered"],
    aliases=["Bundesland","Covid-19 Fälle:",
             "Anzahl der Verstorbenen:","Anzahl der Genesenen:"],
)
```

```
highlight_function = lambda x: {
    'color':"black",
    'weight': 2,

}
```

➡ Zusätzliche Einbindung der Daten zu Verstorbenen und Genesenen

Data Used

4. Zusammenfügen der Daten

Geografischer Mittelpunkt Deutschlands

```
m = folium.Map([51.163409, 10.447718], zoom_start=5.5)

colormap.caption = "Farbskala für die Covid-19 Fälle"
colormap.add_to(m)
```

Legende für die verwendete Skala

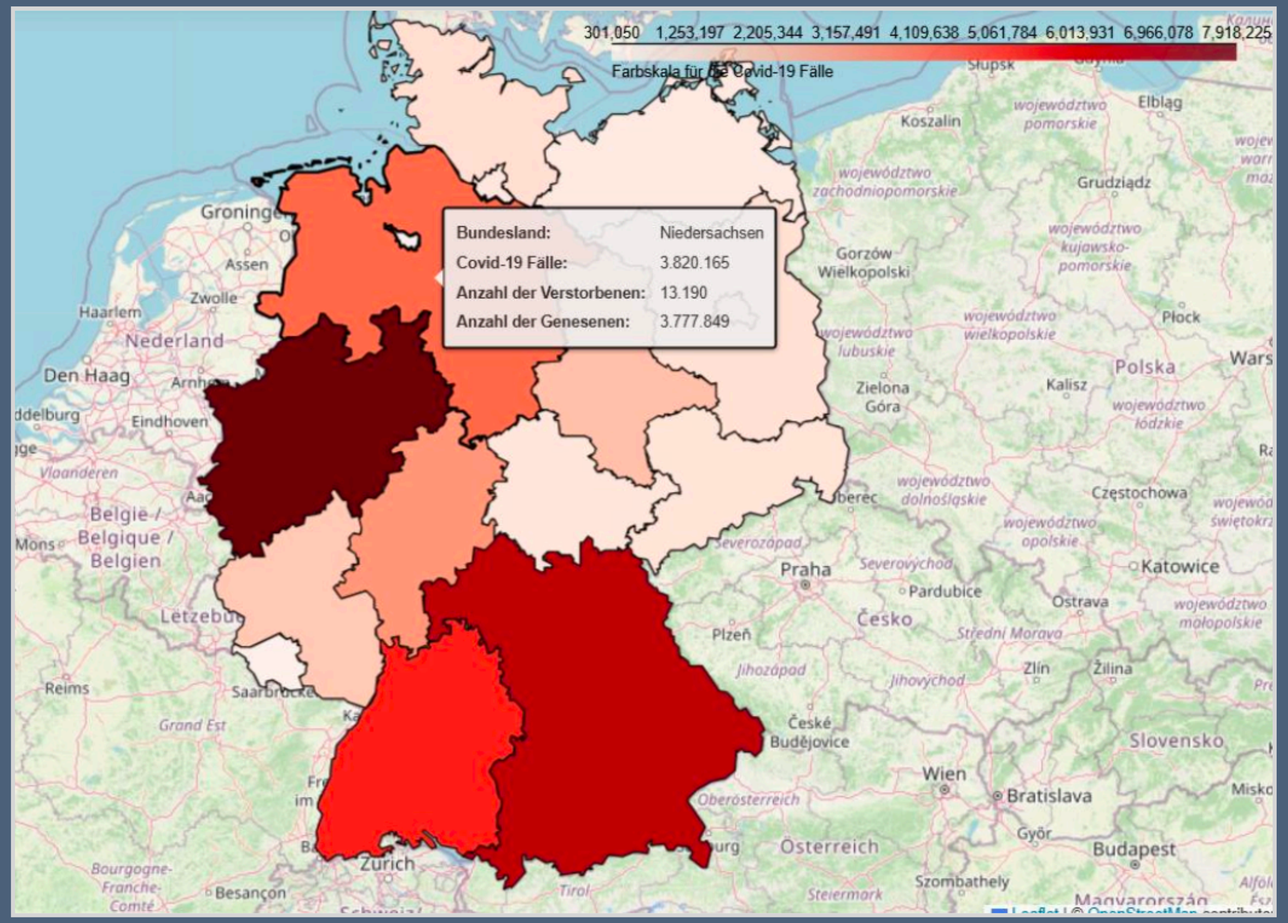
GeoJSON-Daten für die Umrisse

Einbindung der erstellten Farbskala

```
folium.GeoJson(
    bundeslaender,
    zoom_on_click=True,
    name="cases",
    style_function=lambda feature: {
        "fillColor": colormap(data_states_dict[feature["id"]]),
        "color": "black",
        "weight": 1,
        "fillOpacity": 1,
    },
    highlight_function=highlight_function,
    tooltip=tooltip,
    popup=popup,
).add_to(m)
```

Hervorhebungen über Popup, Tooltip und Highlight-Funktion

E2 - Mapping

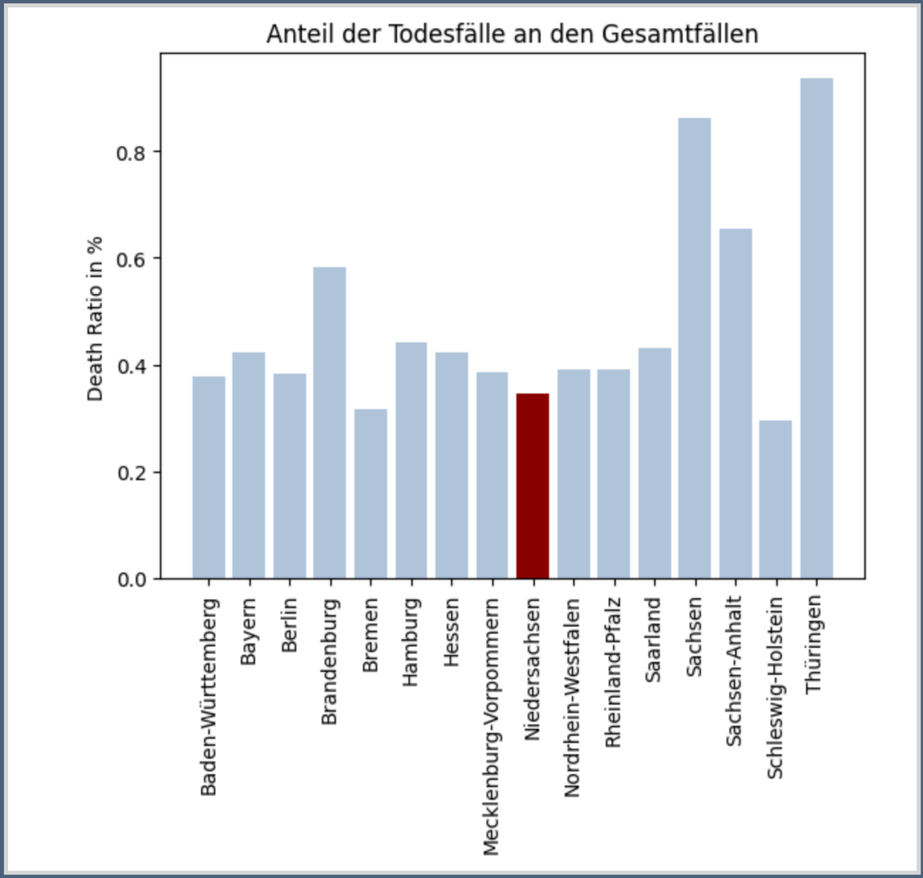


Darstellung der registrierten Covid-19-Fälle nach Bundesländern

Ausschnitt: Popup im Zoom

E2 - Mapping

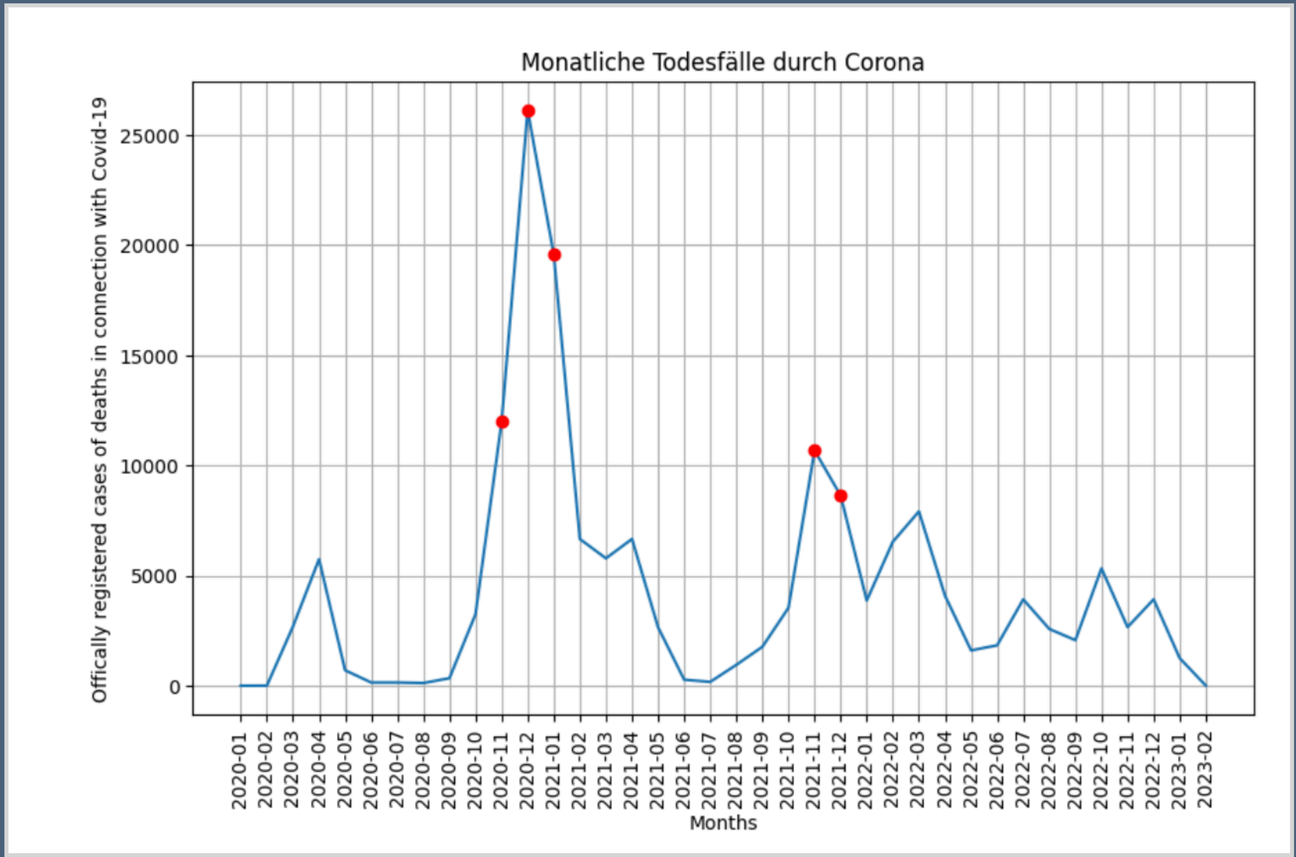
death to cases ratio in Lower Saxony & the period(s) with the most deaths related to Covid-19:



- Bestimmung der Todesdaten als Anteil der Verstorbenen an den Gesamtfällen
- Insgesamt zwischen 0,29% und 0,94%
- Niedersachsen: 0,35%

E2 - Mapping

death to cases ratio in Lower Saxony & the period(s) with the most deaths related to Covid-19:



1) Dezember 2020
2) Januar 2021
3) November 2020
4) November 2021
5) Dezember 2021

- Hypothese: Abhängigkeit von der Jahreszeit (–> Aufgabe 3)

Ranking of States by Recovered COVID-19 Cases

Daten aus der covid_per_state.csv

Metrik zum Vergleich der Genesenen-Zahlen in dt. Bundesländer: Genesenenquote in %

```
# Genesungsquote bestimmen (Genesene-Zahlen/Covid-Fälle)
covid_per_state['genesene_quote'] = (covid_per_state['recovered']/covid_per_state['cases']).replace([float('inf'),
                                    float('-inf')], 0).fillna(0) * 100

# Ranking der Genesungsquoten (abfallend)
covid_per_state_sorted_genesene_quote = covid_per_state.sort_values(by='genesene_quote', ascending=False)

# min. & max. Genesungsquoten berechnen; für die Begrenzung der X-Achse
min_value = covid_per_state_sorted_genesene_quote['genesene_quote'].min()
max_value = covid_per_state_sorted_genesene_quote['genesene_quote'].max()
```

Prozentzahl, die in Abhängigkeit der Infektionszahlen die Effizienz der Genesung zeigt

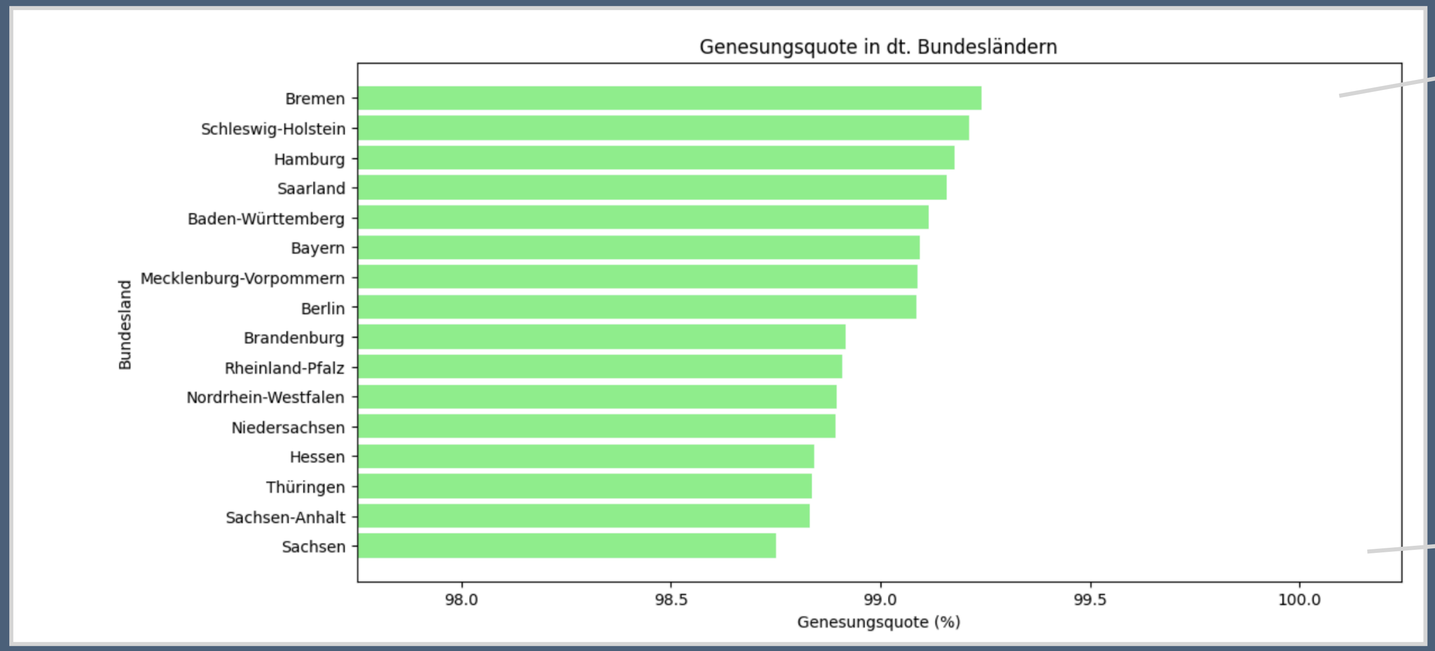
➡ basierend auf dieser Kennzahl: Ranking der Bundesländer entsprechend der höchsten Genesungsrate

Ranking of States by Recovered COVID-19 Cases

Balkendiagramm mit Genesungsquote der dt. Bundesländer

- Hohe Genesungsquote in allen Bunderländern –> minimale Schwankungen mit Prozentzahlen zw. Ca. 98,75 - 99,25 %

Ausgabe:



höchste Genesungsquote: Bremen

Niedrigste Genesungsquote: Sachsen

Schlussfolgerung

- hohe Genesungsquoten deutschlandweit

➡ geringfügige Unterschiede der Bundesländer zurückzuführen auf:

Demografische Unterschiede

Pandemiemanagement

Impfquoten

Gesundheitsinfrastruktur

E3 - Case Studie

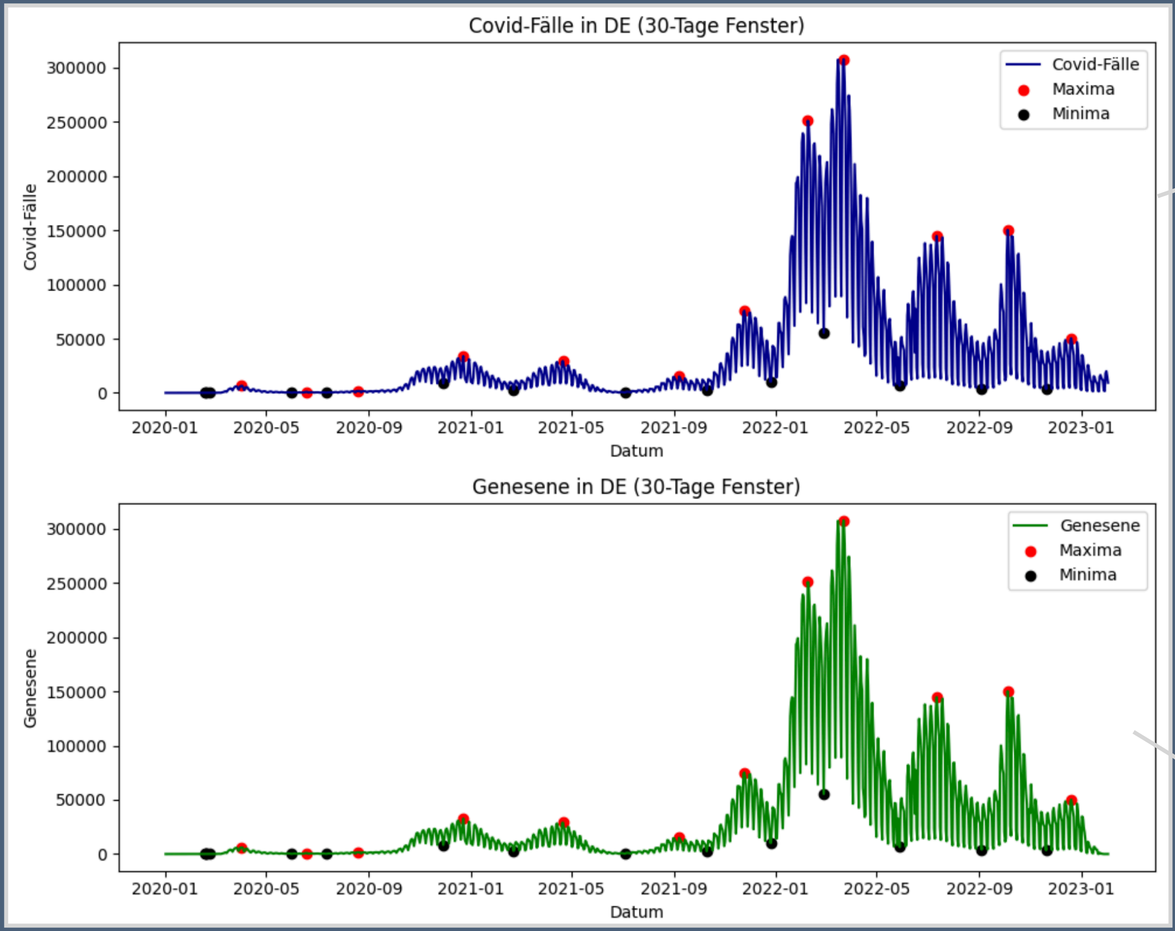
Local maxima and minima of the data set: What conclusion can be drawn from the data?

- Daten aus der covid_de.csv
- Visualisierung der **Covid-Fälle, Genesenen & Todesfälle** in DE im Zeitverlauf (30 Tage Rahmen)

```
# "date"-Daten in DateTime-Format umwandeln
covid_de['date'] = pd.to_datetime(covid_de['date'])

# CoFälle-/Genesene-/Todesfälle-Daten über alle Bundesländer zusammenfügen
covid_de_agg = covid_de.groupby('date')[['cases', 'recovered', 'deaths']].sum().reset_index()

# Lokale Maxima/Minima für alle drei Variablen; 30-Tage Zeitrahmen
covid_de_agg['fälle_lokmax'] = covid_de_agg['cases'] == covid_de_agg['cases'].rolling(window=30, center=True).max()
covid_de_agg['fälle_lokmin'] = covid_de_agg['cases'] == covid_de_agg['cases'].rolling(window=30, center=True).min()

covid_de_agg['genesen_lokmax'] = covid_de_agg['recovered'] == covid_de_agg['recovered'].rolling(window=30,
                                                                                                center=True).max()
covid_de_agg['genesen_lokmin'] = covid_de_agg['recovered'] == covid_de_agg['recovered'].rolling(window=30,
                                                                                                center=True).min()

covid_de_agg['tode_lokmax'] = covid_de_agg['deaths'] == covid_de_agg['deaths'].rolling(window=30, center=True).max()
covid_de_agg['tode_lokmin'] = covid_de_agg['deaths'] == covid_de_agg['deaths'].rolling(window=30, center=True).min()
```

E3 - Case Studie

Local maxima and minima of the data set: What conclusion can be drawn from the data?



Entwicklung der Covid-Fälle: in wellenförmigen Spitzen

1. *Covid-Welle: Frühjahr 2020*
2. *Covid-Welle: Winter 2020/2021*
3. *Covid-Welle: Herbst 2021 bis Anfang 2022*

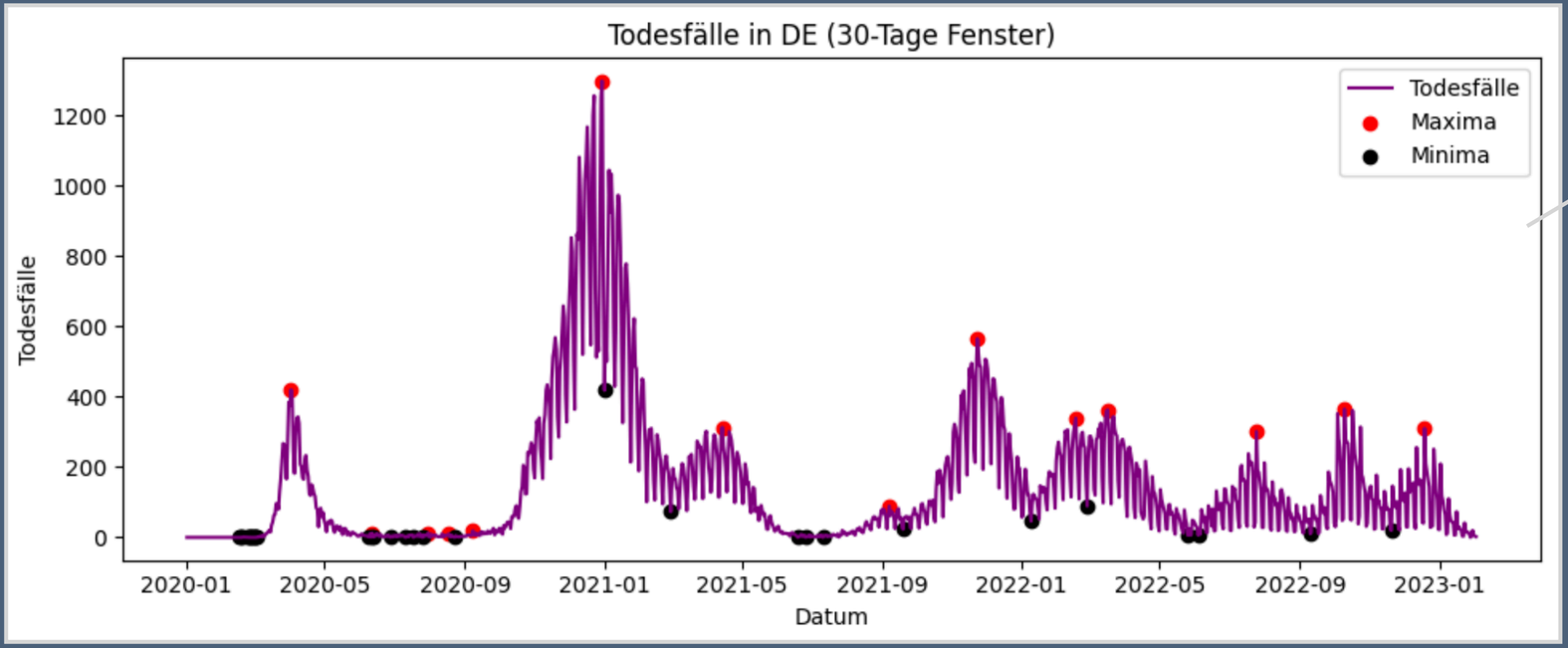
- *erneutes lokales Maximum: Frühjahr/Sommer 2022*

➡ *stärkste Welle: Ende 2021 – Anfang 2022*

Entwicklung der Genesungen:

- *Genesungsverlauf folgt Covid-Fällen mit Verzögerung*
- *größte Zahl an Genesenen während hohen Infektionswellen*
- *hohe Genesungsquote, aber Schwankungen durch Krankheitsverläufe*

Local maxima and minima of the data set: What conclusion can be drawn from the data?



Entwicklung der Todesfälle:

- höchste Sterblichkeit Ende 2020 – Anfang 2021
- Todesfälle sinken im weiteren Verlauf, insbesondere 2022 (trotz hoher Fallzahlen)

Schlussfolgerung

frühere Wellen gefährlicher → höhere Sterblichkeitsrate

- Wintermonate weisen stärkere Virusausbreitung auf → saisonaler Effekt
- Covid-Variante Omikron führte zu höchsten Fallzahlen (bei vgl. geringer Sterblichkeit)
- Schutzmaßnahmen & Impfungen haben Sterblichkeit reduziert

E3 - Case Studie

Does the date of a given datapoint correlate with the number of deaths?

Eingabe:

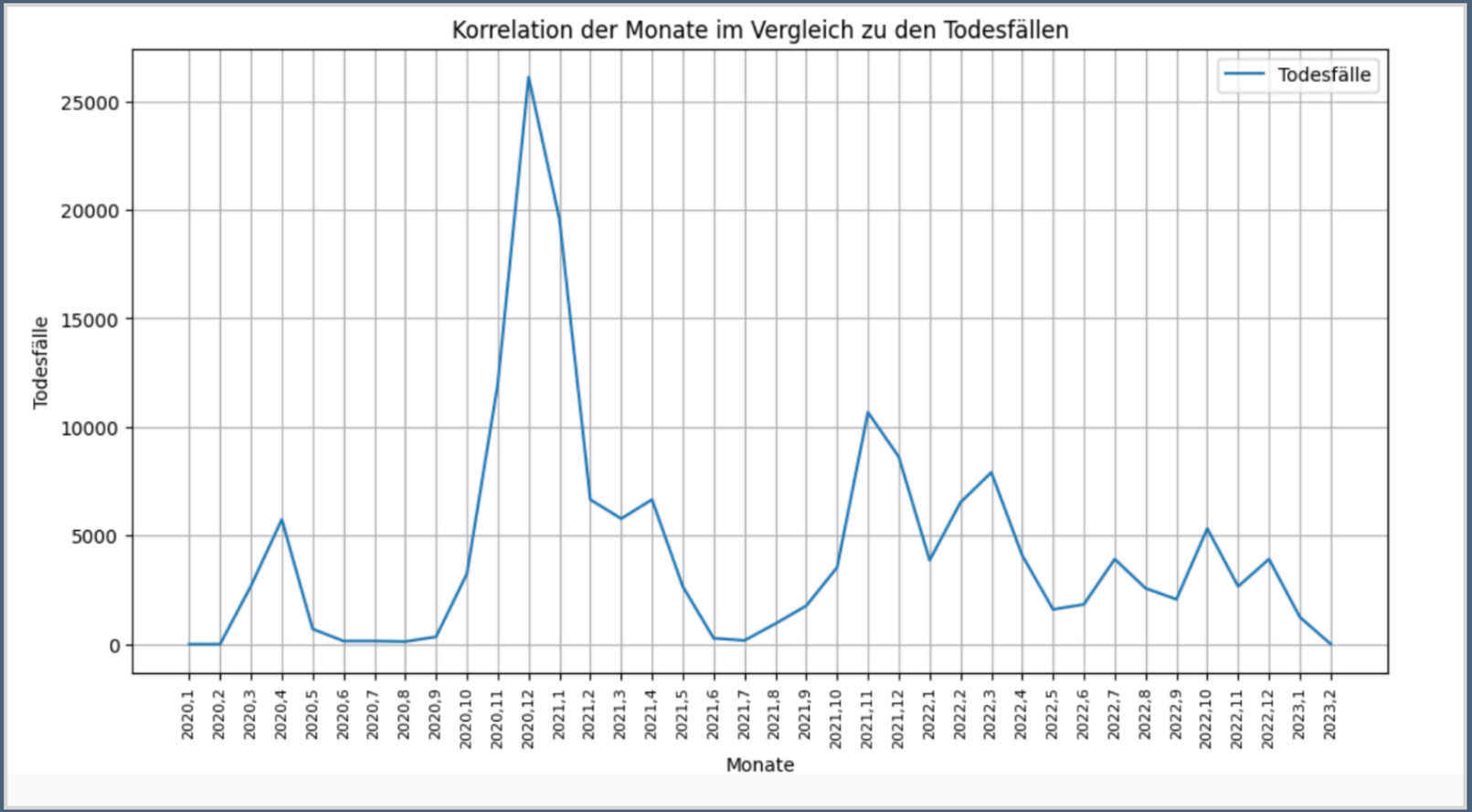
```
#Korrelation berechnen der Monate
spearman_korrelation_months, spearman_p_months = stats.spearmanr(df_covid_agg['year']+ df_covid_agg['month'] / 12,
                                                                 df_covid_agg['deaths'])
print("Die Spearman-Korrelation zwischen den Jahren und den Todesfällen:", spearman_korrelation_months.round(4))
print("Der p-Wert für die Spearman-Korrelation:", spearman_p_months.round(4))
```

Ausgabe:

```
Die Spearman-Korrelation zwischen den Jahren und den Todesfällen: 0.1631
Der p-Wert für die Spearman-Korrelation: 0.328
```

E3 - Case Studie

Does the date of a given datapoint correlate with the number of deaths?



E3 - Case Studie

Does the date of a given datapoint correlate with the number of deaths?

Eingabe:

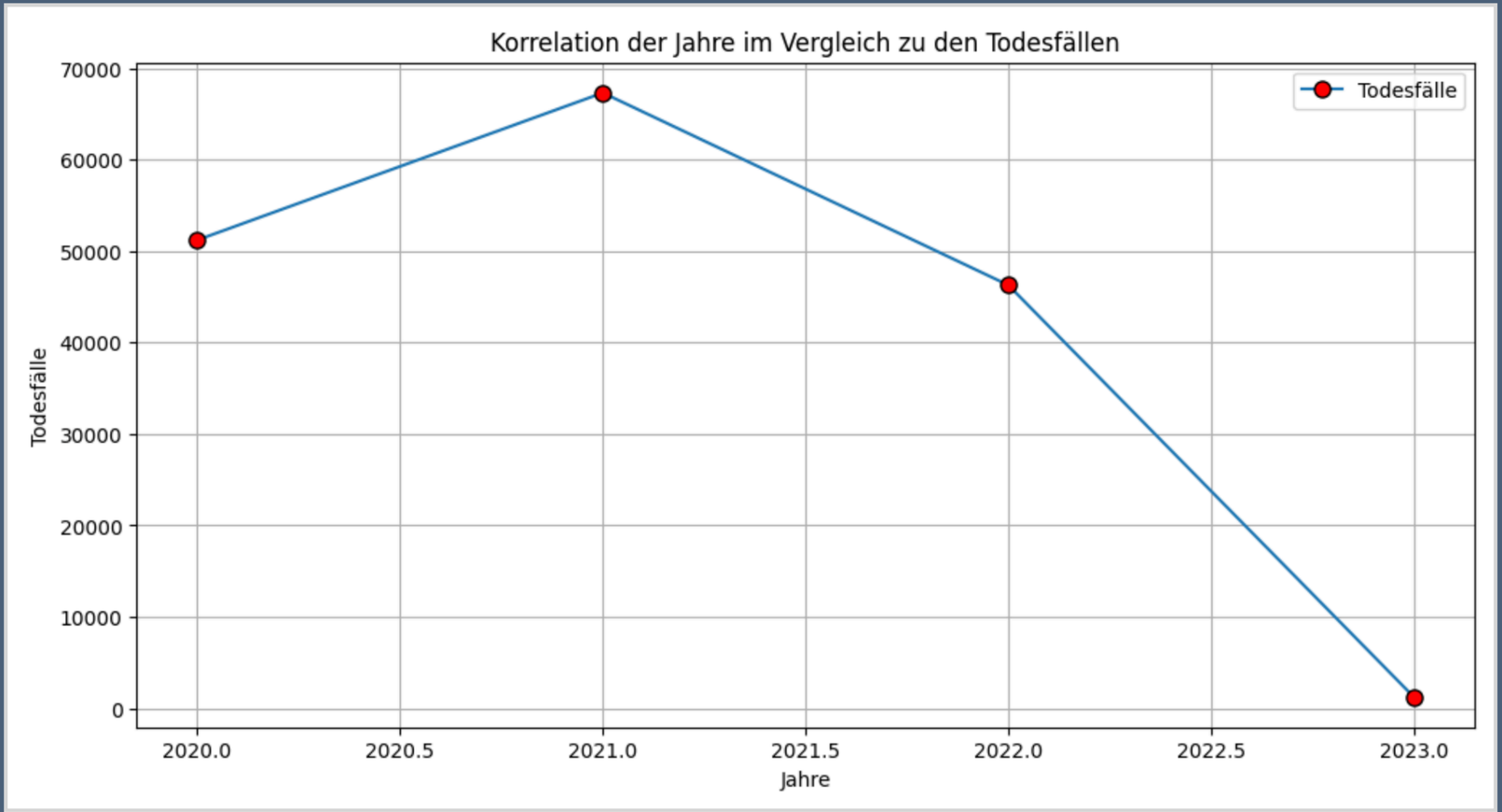
```
#Korrelation berechnen der Jahre
korrelation_years=np.corrcoef(df_yearly_deaths['year'], df_yearly_deaths['deaths'])[0, 1]
print("Die Korrelation zwischen den Jahren und den Todesfällen:",korrelation_years.round(4))
```

Ausgabe:

```
Die Korrelation zwischen den Jahren und den Todesfällen: -0.7795
```

E3 - Case Studie

Does the date of a given datapoint correlate with the number of deaths?



E3 - Case Studie

How likely is it that a Covid-19 infection causes a death/recovery?

Insgesamt:

```
# Datenaufbereitung für die Berechnung der gesamten Wahrscheinlichkeit
df = pd.read_csv('covid_per_state.csv')
case_sum = df['cases'].sum()
rec_sum = df['recovered'].sum()
death_sum = df['deaths'].sum()
p_surv = (rec_sum/case_sum)*100
p_dead = (death_sum/case_sum)*100
print (f"Die Wahrscheinlichkeit an Covid-19-Infekt zu sterben: {p_dead:.4f}%")
print(f"Die Wahrscheinlichkeit an Covid-19-Infekt zu genesen: {p_surv:.4f}%")
```

```
Die Wahrscheinlichkeit an Covid-19-Infekt zu sterben: 0.4391%
Die Wahrscheinlichkeit an Covid-19-Infekt zu genesen: 98.9787%
```

Pro Jahr:

```
# Dataframes erstellen und aufbereiten für die Berechnung der Wahrscheinlichkeiten in den Jahren zu sterben
df_yearly_deaths = df_covid_agg.groupby('year')['deaths'].sum().reset_index()
df_yearly_cases = df_covid.groupby('year')['cases'].sum().reset_index()
df_yearly_recovery = df_covid.groupby('year')['recovered'].sum().reset_index()
df_yearly = pd.merge(df_yearly_cases, df_yearly_deaths, on = 'year')
df_yearly = pd.merge(df_yearly, df_yearly_recovery, on = 'year')
df_yearly['Genesung'] = ((df_yearly['recovered'] / df_yearly['cases'])*100).round(4)
df_yearly['Tod'] = ((df_yearly['deaths'] / df_yearly['cases'])*100).round(4)
```

	year	Tod	Genesung
0	2020	2.9189	97.0811
1	2021	1.2370	98.7627
2	2022	0.1532	99.8459
3	2023	0.2989	47.2563

Fazit

Quellen

E1 - Analyzing the Data

- https://community.databricks.com/t5/data-engineering/how-to-display-markdown-output-in-databricks-notebook-from-a/td-p/32892
- https://www.bundesgesundheitsministerium.de/coronavirus/chronik-coronavirus.html
- https://de.wikipedia.org/wiki/COVID-19-Impfung_in_Deutschland
- https://tradistats.com/exponentieller-gleitender-durchschnitt/
- https://shallbd.com/how-to-calculate-exponential-weighted-moving-average-in-python/
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.ewm.html
- https://numpy.org/doc/stable/reference/generated/numpy.quantile.html
- https://studyflix.de/statistik/quantile-1040
- https://www.rki.de/SharedDocs/FAQs/DE/Impfen/COVID-19/gesamt.html#entry_16869982

Quellen

E1 - Analyzing the Data

- Json load & loads: https://www.geeksforgeeks.org/python-difference-between-json-load-and-json-loads/
- Python GeoJSON: https://tutorpython.com/python-geojson-tutorial-to-read-write-parse-modify
- Datumsinformationen verarbeiten: https://www.data-science-architect.de/datetime-timestamp-in-python/
- Dates in DataFrame: https://luca1iu.medium.com/python-effective-techniques-for-managing-dates-in-dataframe-b64e75c4b777
- pandas.to_datetime: https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html
- Case Fatality Rate (CFR): https://media.sciencemediacenter.de/_legacy/user_upload/Fact_Sheets_PDF/Letalitaet_SARS-CoV-2_SMC_FactSheet_26022020.pdf
- Placing text boxes: https://matplotlib.org/stable/gallery/text_labels_and_annotations/placing_text_boxes.html

Quellen

E2 - Mapping/E3 - Case Studie

- Folium: https://python-visualization.github.io/folium/latest/getting_started.html
- GeoJSON: https://python-visualization.github.io/folium/latest/user_guide/geojson/geojson.html
- Choropleth: https://python-visualization.github.io/folium/latest/user_guide/geojson/choropleth.html
- Colormap: https://python-visualization.github.io/branca/colormap.html
- Pandas: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html, https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.dt.to_period.html, https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html
- Mühlichen, M. (2023). Bevölkerungsforschung Aktuell/ Bundesinstitut für Bevölkerungsforschung, 44,5.S.4-13. https://d-nb.info/1351610902/34
- https://www.bib.bund.de/DE/Service/Veranstaltungen/Berichte/2021/2021-02-25-Berliner-Demografiegespraech-Hoehere-Uebersterblichkeit-in-Deutschland-durch-Corona.html
- https://www.rki.de/DE/Themen/Gesundheit-und-Gesellschaft/Sozialer-Status/Faktenblatt_COVID-19-Sterblichkeit.html
- https://www.geeksforgeeks.org/pearson-vs-spearman-correlation-coefficient/
- https://realpython.com/numpy-scipy-pandas-correlation-python/