Covid19 Germany Deutschland (RKI data)

April 2, 2020

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load in

import os
import urllib
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

# import os
# for dirname, _, filenames in os.walk('/kaggle/input'):
#     for filename in filenames:
#         print(os.path.join(dirname, filename))

# # Any results you write to the current directory are saved as output.

sns.set()
%matplotlib inline

# This checks if the notebook is executed on Kaggle or on your local machine and
# acts accordingly with filenames. On Kaggle the dataset is already managed by␣
↪a managed
# and downloaded to (URLs are linked) a dataset there.
try:
    os.environ['KAGGLE_DATA_PROXY_TOKEN']
except KeyError:
    # daily infections in Germany
    url_daily_corona = "https://opendata.arcgis.com/datasets/
↪dd4580c810204019a7b8eb3e0b329dd6_0.csv"

    # infections per 100k residents in German states
    url_corona_bl_100k = "https://opendata.arcgis.com/datasets/
↪ef4b445a53c1406892257fe63129a8ea_0.csv"

    # infections per 100k residents in German provinces
    url_corona_provinces = "https://opendata.arcgis.com/datasets/
↪917fc37a709542548cc3be077a786c17_0.csv"

    fname1 = "./RKI_covid19_daily_de.csv"
    fname2 = "./RKI_covid19_states_100k.csv"
    fname3 = "./RKI_covid19_provinces_100k.csv"

    urllib.request.urlretrieve(url_daily_corona, fname1)
    urllib.request.urlretrieve(url_daily_corona, fname2)
    urllib.request.urlretrieve(url_daily_corona, fname3)

    fname4 = "./additional_data/covid19_events_measures.csv"

else:
    dirname = "/kaggle/input/covid19-data-germany-robert-koch-institute/"
    dirname_additional = "/kaggle/input/additional-covid19-data/additional_data/
↪"
    # daily infections in Germany
    fname1 = dirname + "dd4580c810204019a7b8eb3e0b329dd6_0.csv"

    # infections per 100k residents in German states
    fname2 = dirname + "ef4b445a53c1406892257fe63129a8ea_0.csv"

    # infections per 100k residents in German provinces
    fname3 = dirname + "917fc37a709542548cc3be077a786c17_0.csv"

    fname4 = dirname_additional + "covid19_events_measures.csv"


df_daily_data = pd.read_csv(fname1, parse_dates=["Meldedatum"],␣
↪date_parser=lambda ts: pd.Timestamp(ts).strftime('%Y-%m-%d'))
df_measures_events = pd.read_csv(fname4, parse_dates=["Date"])

df_daily_data.sort_values(['Meldedatum'], inplace=True)
df_measures_events.sort_values(['Date'], inplace=True)
```

[2]:
```
df_daily_data
```

```
[2]:        IdBundesland          Bundesland            Landkreis Altersgruppe  \
     8827              8  Baden-Württemberg           LK Rastatt       A00-A04
     20329             9             Bayern  LK Landsberg a.Lech       A15-A34
     16140             9             Bayern         LK Starnberg       A35-A59
     26394             9             Bayern           SK München       A15-A34
     20129             9             Bayern  LK Fürstenfeldbruck       A15-A34
     …               …                 …                    …           …
     16610             9             Bayern         SK Straubing       A35-A59
     12757             7    Rheinland-Pfalz  SK Landau i.d.Pfalz       A35-A59
     12761             7    Rheinland-Pfalz  SK Landau i.d.Pfalz       A60-A79
     23169            16          Thüringen     LK Weimarer Land          A80+
     26646             9             Bayern         SK Rosenheim       A60-A79

           Geschlecht  AnzahlFall  AnzahlTodesfall  ObjectId Meldedatum  \
     8827           W           1                0    654601 2020-01-24
     20329          M           1                0    659103 2020-01-28
     16140          M           1                0    659914 2020-01-28
     26394          W           1                0    657789 2020-01-29
     20129          M           1                0    658903 2020-01-29
     …            …           …                …         …          …
     16610          W           5                0    660384 2020-04-01
     12757          W           1                0    651531 2020-04-01
     12761          M           1                0    651535 2020-04-01
     23169          W           1                0    669943 2020-04-01
     26646          W           2                0    658041 2020-04-01

            IdLandkreis         Datenstand  NeuerFall  NeuerTodesfall
     8827          8216  02.04.2020 00:00          0              -9
     20329         9181  02.04.2020 00:00          0              -9
     16140         9188  02.04.2020 00:00          0              -9
     26394         9162  02.04.2020 00:00          0              -9
     20129         9179  02.04.2020 00:00          0              -9
     …            …               …              …             …
     16610         9263  02.04.2020 00:00          1              -9
     12757         7313  02.04.2020 00:00          1              -9
     12761         7313  02.04.2020 00:00          1              -9
     23169        16071  02.04.2020 00:00          1              -9
     26646         9163  02.04.2020 00:00          1              -9

     [30379 rows x 13 columns]
```

```
[3]: print("Measures taken and events/incidents concerning Covid19."
           + "These might show a connection to the falling or rising of the curves:")
     df_measures_events
```

```
Measures taken and events/incidents concerning Covid19.These might show a
connection to the falling or rising of the curves:
```

```
[3]:         Date                                       In_Short   State  \
     0 2020-03-16                Schools, day care centers closed.    Alle
     1 2020-03-16             Entry bans and strict border controls.    Alle
     2 2020-03-20     Quarantine Bavaria, only vital locations open.  Bayern
     3 2020-03-22  Quarantine country-wide, only vital locations …    Alle

                                            Description
     0                   Schools, day care centers closed.
     1  Germany has decided on entry bans and strict c…
     2  Only vital location, like supermarkets, doctor…
     3  Only vital location, like supermarkets, doctor…
```

```
[4]: print("Total sum of cases in Germany:")
     df_daily_data["AnzahlFall"].sum()
```

```
Total sum of cases in Germany:
```

```
[4]: 73455
```

```
[5]: print("Describe number of cases per day in Germany:")
     df_daily_data.groupby("Meldedatum").sum()["AnzahlFall"].describe()
```

```
Describe number of cases per day in Germany:
```

```
[5]: count      51.000000
     mean     1440.294118
     std      1911.555579
     min         1.000000
     25%         5.000000
     50%       176.000000
     75%      3018.000000
     max      5957.000000
     Name: AnzahlFall, dtype: float64
```

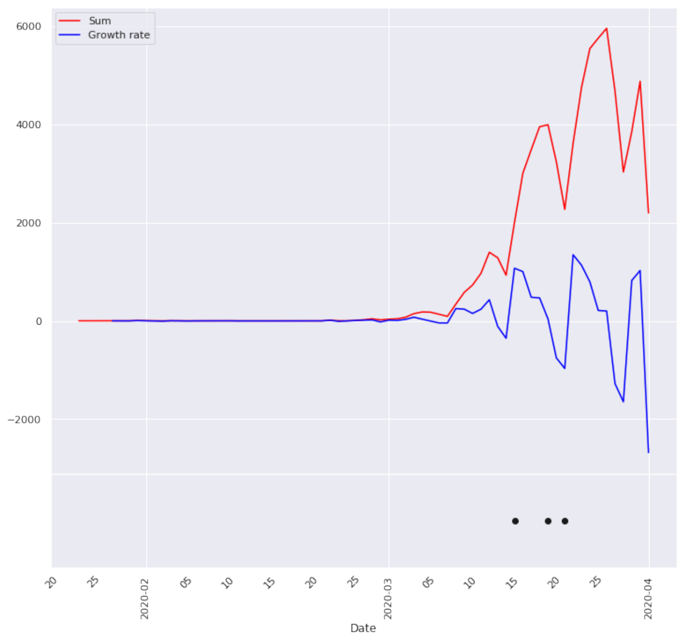
```
[6]: # def annotationsForDate(df, date):
     #     """
     #     Takes the measures df and a time index and returns all the short
     #     measure descriptions as a string for the annotations below
     #     """
     #     query = df_measures_events.query("Date == '{}'".format(date))
     #     ann = ""
     #     for i in query.index:
     #         state = ""
     #         if query["State"].iloc[i]:
     #             state = query["State"].iloc[i] + ": "
     #         s = query["In_Short"].iloc[i]
     #         ann += s + '\n'
#     return ann.rstrip()
```

```
[7]: gr_day_reported = df_daily_data.groupby('Meldedatum')

     # TODO: make these relative to screen size somehow
     figsize = (12,11)
     fig, ax = plt.subplots(2, 1, figsize=figsize, sharex=True,␣
      ↪gridspec_kw={'height_ratios': [5, 1]})
     fig.subplots_adjust(hspace=0.0)
     suptitel_attr = {"fontsize" : 16, "fontweight" : "bold", "ha" : "center", "va" :
      ↪ "bottom", "y" : 0.94}

     # set plot stuff
     fig.suptitle("Sum of confirmed cases per day and growth rate thereof for␣
      ↪Germany. Below are points "
                  + "for political measures or events/incidents.", **suptitel_attr)

     for axis in ax.flat:
         axis.set_xlabel('Date')
         rule = mpl.dates.rrulewrapper(mpl.dates.MONTHLY,  bymonthday=(5, 10, 15,␣
      ↪20, 25))
         axis.xaxis.set_minor_locator(mpl.dates.RRuleLocator(rule))
         axis.xaxis.set_minor_formatter(mpl.dates.DateFormatter("%d"))
         axis.xaxis.set_major_locator(mpl.dates.MonthLocator(bymonthday=1))
         axis.xaxis.set_major_formatter(mpl.dates.DateFormatter('%Y-%m'))

     ax[1].yaxis.set_minor_locator(mpl.ticker.NullLocator())
     ax[1].yaxis.set_major_locator(mpl.ticker.NullLocator())

     sum_cases_per_day = gr_day_reported['AnzahlFall'].sum()
     growth_rate = sum_cases_per_day.diff()

     ax[0].plot(sum_cases_per_day, color='red', label='Sum')
     ax[0].plot(growth_rate, color='blue', label='Growth rate')

     plt.setp(ax[0].xaxis.get_majorticklabels(), rotation=90)
     plt.setp(ax[0].xaxis.get_minorticklabels(), rotation=45)


     for d in df_measures_events["Date"].unique():
         #s = annotationsForDate(df_measures_events, d)
         #axs[1].annotate(s = "bla", xy=(d, sum_cases_per_day[d]), xytext=(-50,-10),␣
      ↪xycoords="data")
         pass

     ax[1].scatter(df_measures_events["Date"].unique(),␣
      ↪len(df_measures_events["Date"].unique()) * [100], c='k')
```

```
for axis in ax.flat:
    plt.setp(axis.xaxis.get_majorticklabels(), rotation=90)
    plt.setp(axis.xaxis.get_minorticklabels(), rotation=45)

ax[0].legend()
```

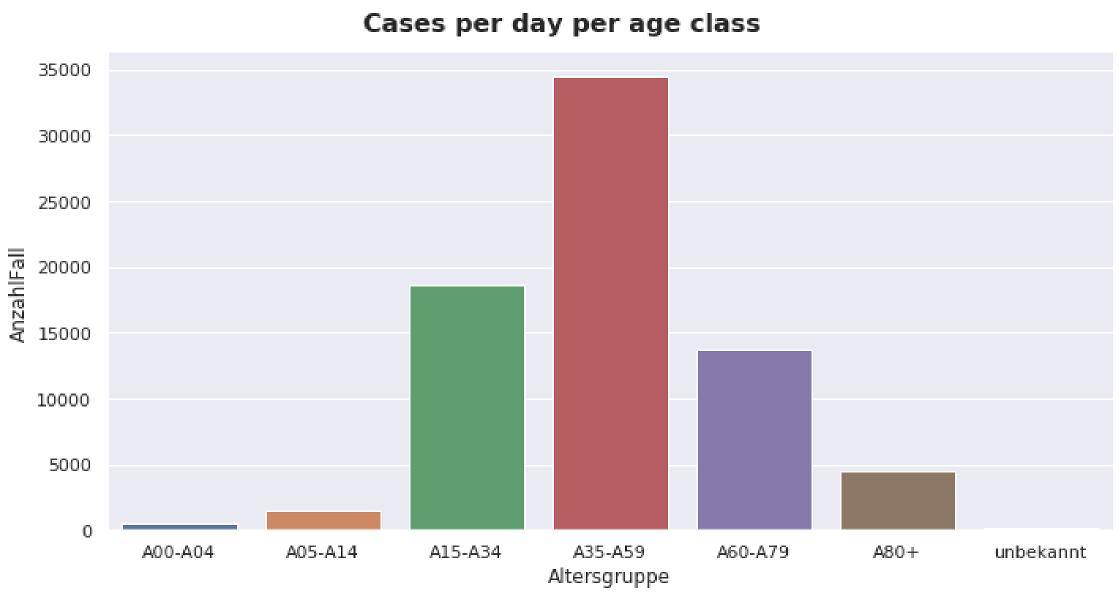
```
[7]: <matplotlib.legend.Legend at 0x7f829b6c0890>
```

Sum of confirmed cases per day and growth rate thereof for Germany. Below are points for political measures or events/incidents.



```
[8]: g_ages = df_daily_data.groupby("Altersgruppe")
ages_cases = g_ages["AnzahlFall"].sum().reset_index()
cp = sns.catplot("Altersgruppe", "AnzahlFall", data=ages_cases, kind="bar",␣
 ↪aspect = 2)
suptitel_attr = {"fontsize" : 16, "fontweight" : "bold", "ha" : "center", "va" :
 ↪ "bottom", "y" : 1}
cp.fig.suptitle("Cases per day per age class", **suptitel_attr)
```

```
[8]: Text(0.5, 1, 'Cases per day per age class')
```



[9]:
```
print("Total sum of cases per German state")
df_daily_data.groupby(['Bundesland'])["AnzahlFall"].sum()
```

```
Total sum of cases per German state
```

[9]:
```
Bundesland
Baden-Württemberg          14647
Bayern                     18489
Berlin                      2968
Brandenburg                  993
Bremen                       327
Hamburg                     2404
Hessen                      3705
Mecklenburg-Vorpommern       438
Niedersachsen               4687
Nordrhein-Westfalen        15415
Rheinland-Pfalz             3118
Saarland                    1020
Sachsen                     2180
Sachsen-Anhalt               804
Schleswig-Holstein          1335
Thüringen                    925
Name: AnzahlFall, dtype: int64
```

[10]:
```
print("Describe sum of daily cases per day per German state.")
df_daily_data.groupby(['Bundesland', "Meldedatum"])['AnzahlFall'].sum().
 ↪unstack().T.fillna(0).describe().round(decimals=0)
```

```
Describe sum of daily cases per day per German state.
```

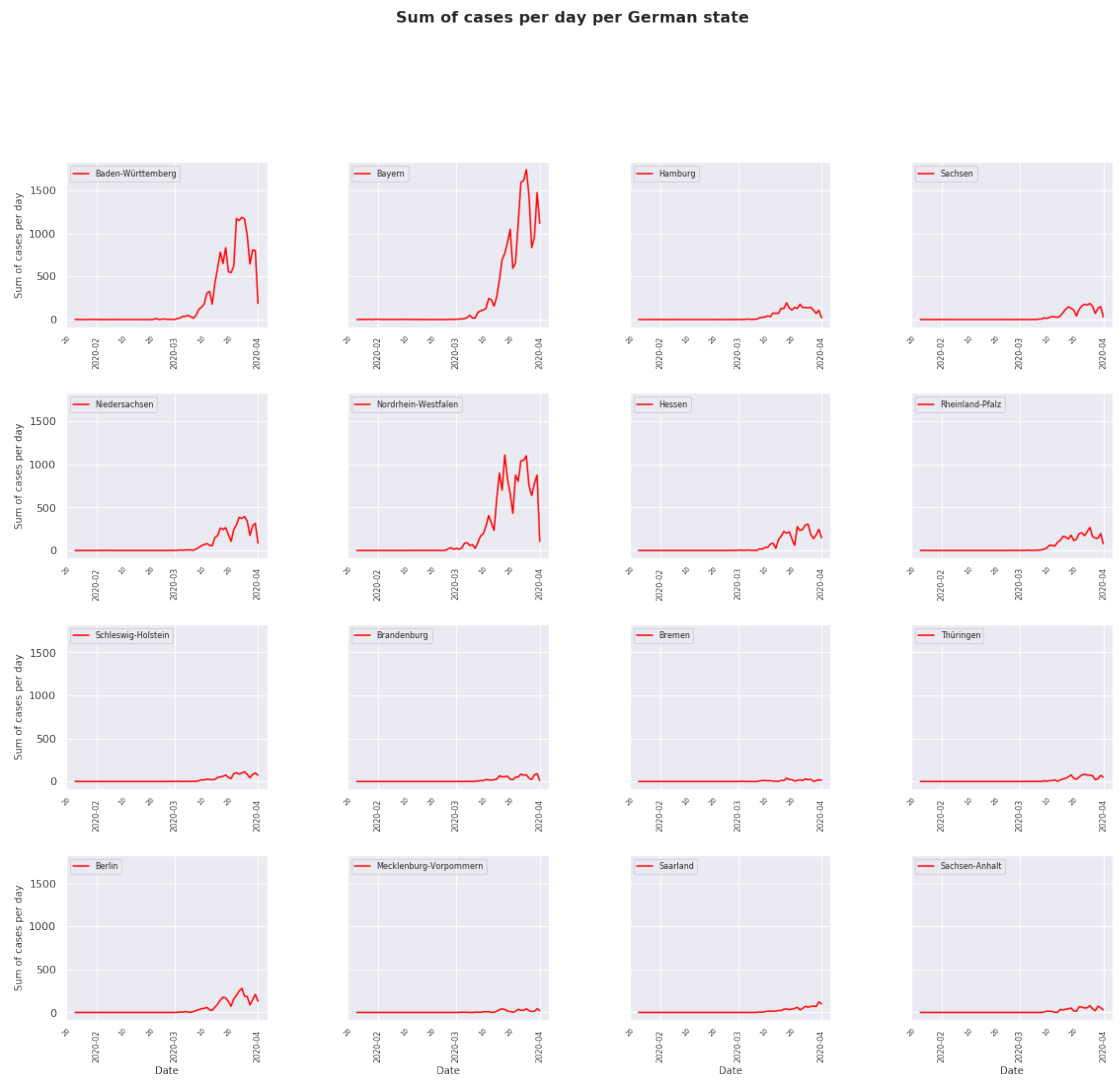
[10]:
```
Bundesland  Baden-Württemberg  Bayern  Berlin  Brandenburg  Bremen  Hamburg  \
count                    51.0    51.0    51.0         51.0    51.0     51.0
mean                    287.0   363.0    58.0         19.0     6.0     47.0
std                     388.0   531.0    80.0         28.0     9.0     61.0
min                       0.0     0.0     0.0          0.0     0.0      0.0
25%                       2.0     2.0     0.0          0.0     0.0      0.0
50%                      37.0    22.0    11.0          1.0     1.0      5.0
75%                     578.0   678.0   112.0         26.0    11.0    108.0
max                    1189.0  1743.0   280.0         92.0    40.0    194.0

Bundesland  Hessen  Mecklenburg-Vorpommern  Niedersachsen  \
count         51.0                    51.0           51.0
mean          73.0                     9.0           92.0
std           99.0                    13.0          128.0
min            0.0                     0.0            0.0
25%            0.0                     0.0            0.0
50%            4.0                     1.0            7.0
75%          142.0                    13.0          174.0
max          305.0                    44.0          396.0

Bundesland  Nordrhein-Westfalen  Rheinland-Pfalz  Saarland  Sachsen  \
count                      51.0             51.0      51.0     51.0
mean                      302.0             61.0      20.0     43.0
std                       380.0             80.0      30.0     61.0
min                         0.0              0.0       0.0      0.0
25%                         0.0              0.0       0.0      0.0
50%                        83.0              3.0       0.0      2.0
75%                       649.0            132.0      36.0     76.0
max                      1111.0            268.0     122.0    186.0

Bundesland  Sachsen-Anhalt  Schleswig-Holstein  Thüringen
count                 51.0                51.0       51.0
mean                  16.0                26.0       18.0
std                   23.0                36.0       27.0
min                    0.0                 0.0        0.0
25%                    0.0                 0.0        0.0
50%                    0.0                 2.0        0.0
75%                   30.0                46.0       31.0
max                   78.0               113.0       83.0
```

[11]:
```python
# Get daily cases and growth rate per German state

loc_unknown_col = "-nicht erhoben-"
state_names = df_daily_data['Bundesland'].unique()
# remove column name for unknown location of the case
```

```
index = np.where(state_names == loc_unknown_col)
state_names = np.delete(state_names, index)

states_daily = df_daily_data.groupby(['Meldedatum',␣
 ↪'Bundesland'])['AnzahlFall'].sum()
df_per_state_daily = pd.DataFrame()
for state in state_names:
    df_per_state_daily[state] = states_daily.unstack()[state].fillna(0)
```

[12]:
```
# TODO: make these relative to screen size somehow
figsize = (19,16)
ax_label_fontsize = 10.0
legend_fontsize = 8.0
major_tick_fontsize = 8.0
minor_tick_fontsize = 7.0
tick_monthdays = (10, 20)
```

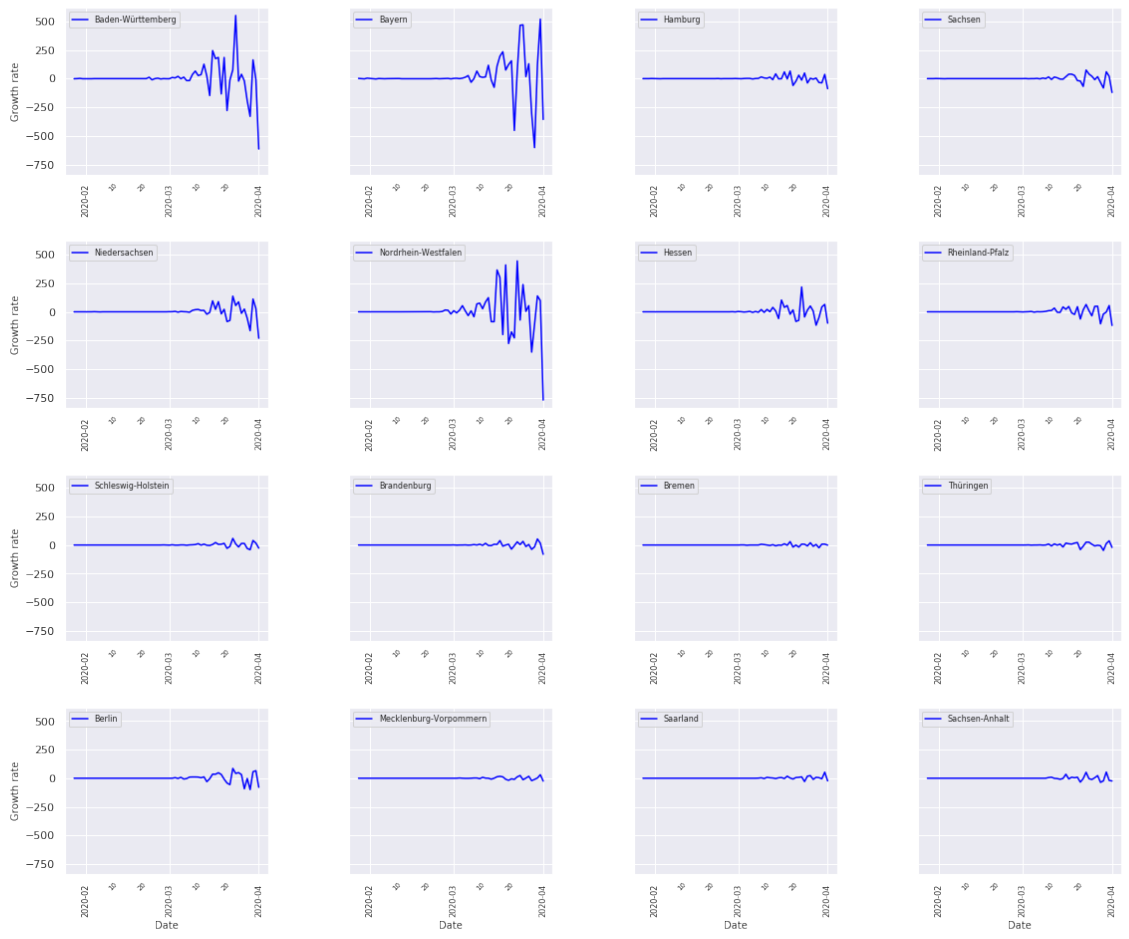
[13]:
```
# Plot sum of confirmed cases per day per German state

fig, ax = plt.subplots(4, 4, sharey=True, figsize=figsize)
fig.subplots_adjust(hspace = 0.4, wspace = 0.4)

fig.suptitle("Sum of cases per day per German state", **suptitel_attr)

for axis in ax.flat:
    rule = mpl.dates.rrulewrapper(mpl.dates.MONTHLY,  bymonthday=tick_monthdays)
    axis.xaxis.set_minor_locator(mpl.dates.RRuleLocator(rule))
    axis.xaxis.set_minor_formatter(mpl.dates.DateFormatter("%d"))
    axis.xaxis.set_major_locator(mpl.dates.MonthLocator(bymonthday=1))
    axis.xaxis.set_major_formatter(mpl.dates.DateFormatter('%Y-%m'))

    for tick in axis.xaxis.get_major_ticks():
            tick.label.set_fontsize(tick_fontsize)

for plt_row in ax:
    plt_row[0].set_ylabel('Sum of cases per day', fontsize=ax_label_fontsize)

for column_plots in ax.T:
    column_plots[-1].set_xlabel('Date', fontsize=ax_label_fontsize)

ix = 0
for subp in ax.flat:
    state = state_names[ix]
    subp.plot(df_per_state_daily[state], color='red', label = "{}".
 ↪format(state))
    subp.legend(fontsize=legend_fontsize, loc="upper left")
    plt.setp(subp.xaxis.get_majorticklabels(), rotation=90)
    plt.setp(subp.xaxis.get_minorticklabels(), rotation=45)
    for tick in subp.xaxis.get_major_ticks():
            tick.label.set_fontsize(major_tick_fontsize)
    for tick in subp.xaxis.get_minor_ticks():
            tick.label.set_fontsize(minor_tick_fontsize)
    ix += 1
```



[14]:
```
# Plot growth rate of daily confirmed cases per German state

fig, ax = plt.subplots(4, 4, sharey=True, figsize=figsize)
fig.subplots_adjust(hspace = 0.4, wspace = 0.4)

fig.suptitle("Growth rate of cases per day per German state", **suptitel_attr)
```

```
for axis in ax.flat:
    rule = mpl.dates.rrulewrapper(mpl.dates.MONTHLY,  bymonthday=tick_monthdays)
    axis.xaxis.set_minor_locator(mpl.dates.RRuleLocator(rule))
    axis.xaxis.set_minor_formatter(mpl.dates.DateFormatter("%d"))
    axis.xaxis.set_major_locator(mpl.dates.MonthLocator(bymonthday=1))
    axis.xaxis.set_major_formatter(mpl.dates.DateFormatter('%Y-%m'))

    for tick in axis.xaxis.get_major_ticks():
            tick.label.set_fontsize(tick_fontsize)

for plt_row in ax:
    plt_row[0].set_ylabel('Growth rate', fontsize=ax_label_fontsize)

for column_plots in ax.T:
    column_plots[-1].set_xlabel('Date', fontsize=ax_label_fontsize)

ix = 0
for subp in ax.flat:
    state = state_names[ix]
    subp.plot(df_per_state_daily[state].diff(), color='blue', label = "{}".
 ↪format(state))
    subp.legend(fontsize=legend_fontsize, loc="upper left")
    plt.setp(subp.xaxis.get_majorticklabels(), rotation=90)
    plt.setp(subp.xaxis.get_minorticklabels(), rotation=45)
    for tick in subp.xaxis.get_major_ticks():
            tick.label.set_fontsize(major_tick_fontsize)
    for tick in subp.xaxis.get_minor_ticks():
            tick.label.set_fontsize(minor_tick_fontsize)
    ix += 1
```

Growth rate of cases per day per German state


Baden-Württemberg
Bayern
Hamburg
Sachsen
Niedersachsen
Nordrhein-Westfalen
Hessen
Rheinland-Pfalz
Schleswig-Holstein
Brandenburg
Bremen
Thüringen
Berlin
Mecklenburg-Vorpommern
Saarland
Sachsen-Anhalt
Growth rate
Date