Covid-19 Vaccination Campaign in Germany

The data used here were provided by Robert Koch Institute and the German federal ministry of Health.

These institutions publish the datasets and some analysis on the page impfdashboard.de.

Setup

Imports

In [1]:
```
# standard library
import datetime
import math
```

In [2]:
```
# third party
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
```

Date this Notebook was run

In [3]:
```
today = datetime.datetime.today().strftime('%Y-%m-%d')
today
```

Out[3]:
```
'2021-04-21'
```

Set Defaults

In [4]:
```
# style like ggplot in R
plt.style.use('ggplot')
```

Get and Transform Data

In [5]:
```
vaccination_data_permalink = 'https://impfdashboard.de/static/data/germany_vaccinations_timeseries_v2.tsv'
vaccinations = pd.read_csv(
    vaccination_data_permalink,
    sep="\t")
```

Drop unnecessary columns

Columns with names starting with 'indikation_' will not be analyzed as the data providers stopped updating them.

In [6]:
```
# No analysis of indication planned:
cols_to_drop = vaccinations.columns[vaccinations.columns.str.contains('indikation_')]
vaccinations.drop(columns=cols_to_drop, inplace=True)
```

In [7]:
```
# Convert datatype of date column
vaccinations.iloc[ : , [0]] = vaccinations.iloc[ : , [0]].apply(pd.to_datetime)
```

Show Data

In [8]:
```
vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115 entries, 0 to 114
Data columns (total 12 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   date                              115 non-null    datetime64[ns]
 1   dosen_kumulativ                   115 non-null    int64
 2   dosen_differenz_zum_vortag        115 non-null    int64
 3   dosen_erst_differenz_zum_vortag   115 non-null    int64
 4   dosen_zweit_differenz_zum_vortag  115 non-null    int64
 5   dosen_biontech_kumulativ          115 non-null    int64
 6   dosen_moderna_kumulativ           115 non-null    int64
 7   dosen_astrazeneca_kumulativ       115 non-null    int64
 8   personen_erst_kumulativ           115 non-null    int64
 9   personen_voll_kumulativ           115 non-null    int64
 10  impf_quote_erst                   115 non-null    float64
 11  impf_quote_voll                   115 non-null    float64
dtypes: datetime64[ns](1), float64(2), int64(9)
memory usage: 10.9 KB
```

In [9]:
```
vaccinations.tail(3)
```

Out[9]:

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
112	2021-04-18	22015018	261476	223153	38323	1626127
113	2021-04-19	22400394	385376	326894	58482	1654137
114	2021-04-20	22935592	535198	477320	57878	1690507

Last Update

Often the data is not updated on weekends, so get the highest date in the dataset.

```
In [10]: last_update = vaccinations.loc[vaccinations.index[-1], "date"].strftime('%Y-%m-%d')
         last_update
```

```
Out[10]: '2021-04-20'
```

Doses Used

```
In [11]: doses = vaccinations.loc[ : , ['date', 'dosen_differenz_zum_vortag']]
         # Rename columns
         doses.columns = ['date', 'doses used']
```

```
In [12]: # Scale number of doses as millions
         doses['doses used'] = doses['doses used'] / 1_000_000
```

Doses Daily

```
In [13]: doses_daily = doses.set_index('date', inplace=False)
         doses_daily.tail(1)
```

Out[13]:

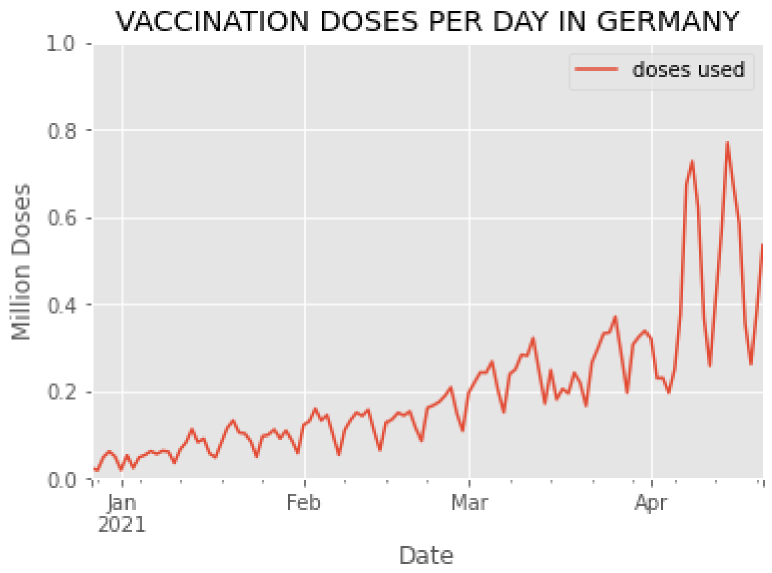
	doses used
date	
2021-04-20	0.535198

```
In [14]: # What is the highest number of doses used in a day?
         max_doses_daily = max(doses_daily['doses used'])
         max_doses_daily
```

```
Out[14]: 0.770046
```

```
In [15]: doses_daily.plot(
             ylim=(0,math.ceil(max_doses_daily)),
             xlabel='Date',
             ylabel='Million Doses',
             title='VACCINATION DOSES PER DAY IN GERMANY')
```

```
Out[15]: <AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>
```



Doses per Weekday (in the last 6 weeks)

```
In [16]: last_6_weeks = doses.tail(42)
```

```
In [17]: # Yields a warning, but exactly like the docs prescribe and it works
         # https://pandas.pydata.org/docs/getting_started/intro_tutorials/05_add_columns.html
         last_6_weeks['weekday'] = last_6_weeks['date'].dt.day_name()
```

```
<ipython-input-17-45013977109e>:3: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  last_6_weeks['weekday'] = last_6_weeks['date'].dt.day_name()
```

In [18]:
```
# check:
last_6_weeks.tail(3)
```

Out[18]:

	date	doses used	weekday
112	2021-04-18	0.261476	Sunday
113	2021-04-19	0.385376	Monday
114	2021-04-20	0.535198	Tuesday

In [19]:
```
# drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

In [20]:
```
#last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

Out[20]:

	doses used	weekday
112	0.261476	Sunday
113	0.385376	Monday
114	0.535198	Tuesday

In [21]:
```
pivot_table =last_6_weeks.pivot(columns='weekday', values='doses used')
pivot_table.tail()
```

Out[21]:

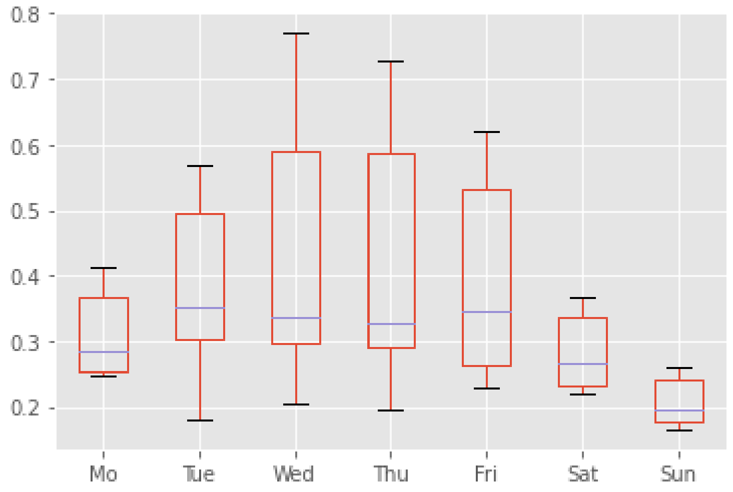
weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
110	0.584797	NaN	NaN	NaN	NaN	NaN	NaN
111	NaN	NaN	0.355228	NaN	NaN	NaN	NaN
112	NaN	NaN	NaN	0.261476	NaN	NaN	NaN
113	NaN	0.385376	NaN	NaN	NaN	NaN	NaN
114	NaN	NaN	NaN	NaN	NaN	0.535198	NaN

In [22]:
```
# Reorder the columns
pivot_table = pivot_table[['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']]
# Rename the columns
pivot_table.columns=['Mo', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
pivot_table.tail()
```

Out[22]:

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
110	NaN	NaN	NaN	NaN	0.584797	NaN	NaN
111	NaN	NaN	NaN	NaN	NaN	0.355228	NaN
112	NaN	NaN	NaN	NaN	NaN	NaN	0.261476
113	0.385376	NaN	NaN	NaN	NaN	NaN	NaN
114	NaN	0.535198	NaN	NaN	NaN	NaN	NaN

In [23]:
```
weekday_boxplot = pivot_table.boxplot()
```



In [24]:
```
fig = weekday_boxplot.get_figure()
fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

In [25]:
```
# W-Mon in order to start the week on a Monday, see:
```

```
# https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#anchored-offsets
doses_weekly = doses.groupby(pd.Grouper(key='date',freq='W-Mon')).sum()
doses_weekly.columns = ['million doses used']
doses_weekly.tail()
```

Out[25]:

date	million doses used
2021-03-29	2.121921
2021-04-05	1.887451
2021-04-12	3.440831
2021-04-19	3.599030
2021-04-26	0.535198

In [26]:

```
# What is the highest number of doses used in a week?
max_million_doses_weekly = max(doses_weekly['million doses used'])
max_million_doses_weekly
```

Out[26]:

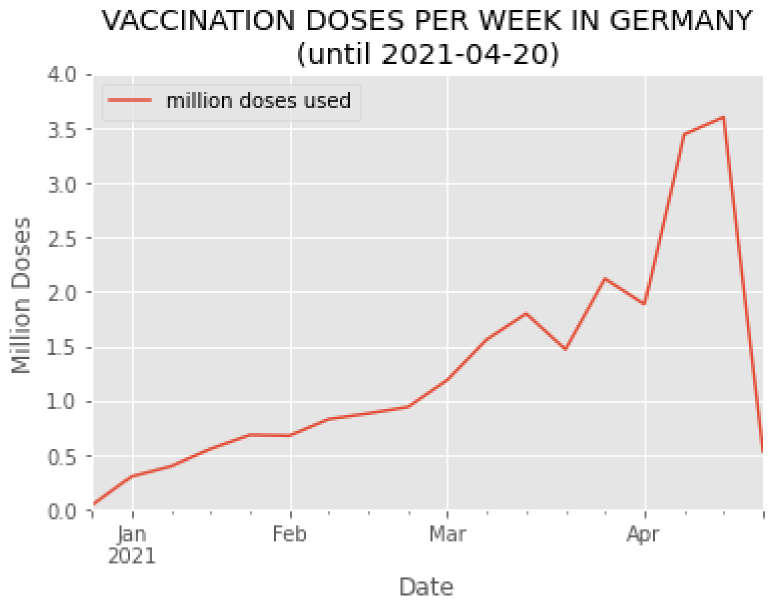
```
3.5990299999999995
```

In [27]:

```
doses_weekly.plot(
    ylim=(0, math.ceil(max_million_doses_weekly)),
    xlabel='Date',
    ylabel='Million Doses',
    title=f"VACCINATION DOSES PER WEEK IN GERMANY\n(until {last_update})")
```

Out[27]:

```
<AxesSubplot:title={'center':'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-04-20)'}, xlabel='Date', ylabel='Milli
on Doses'>
```



Doses per Month

In [28]:
```
# M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date',freq='M')).sum()
doses_monthly.tail()
```

Out[28]:

date	doses used
2020-12-31	0.203970
2021-01-31	2.343276
2021-02-28	3.772680
2021-03-31	7.815962
2021-04-30	8.799704

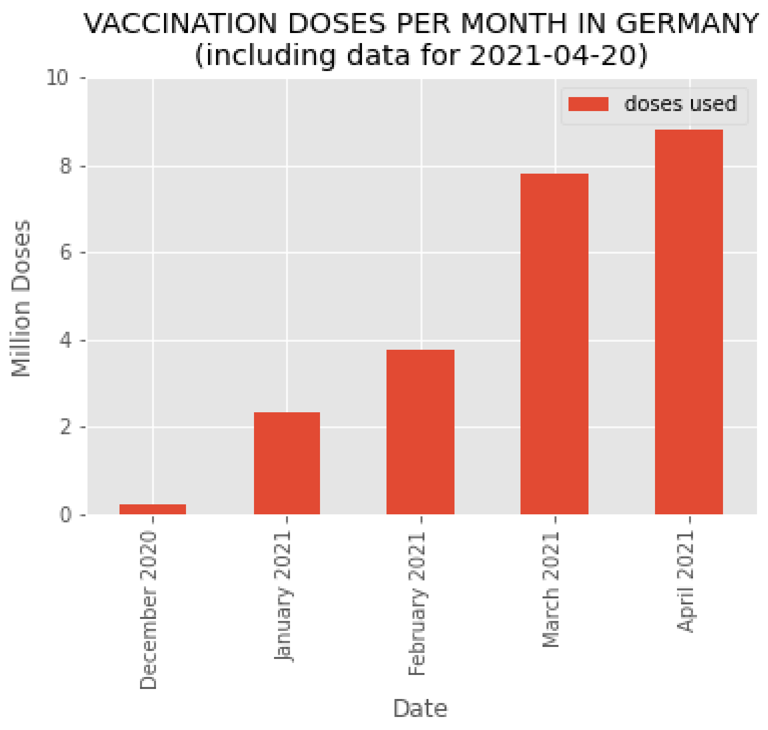
In [29]:
```
max_doses_monthly = max(doses_monthly['doses used'])
max_doses_monthly
doses_monthly['month'] = doses_monthly.index.strftime('%B')
doses_monthly['year'] =  doses_monthly.index.strftime('%Y')
```

```
doses_monthly['label'] = doses_monthly['month'] + ' ' + doses_monthly['year']
doses_monthly.drop(columns=['month', 'year'], inplace=True)
doses_monthly.set_index('label', inplace=True)
doses_monthly.tail(6)
```

Out[29]:

	doses used
label	
December 2020	0.203970
January 2021	2.343276
February 2021	3.772680
March 2021	7.815962
April 2021	8.799704

In [30]:
```
monthly_plot = doses_monthly.plot.bar(
    ylim=(0,math.ceil(max_doses_monthly) + 1),
    xlabel='Date',
    ylabel='Million Doses',
    title=f"VACCINATION DOSES PER MONTH IN GERMANY\n(including data for {last_update})")
```



```
In [31]: fig = monthly_plot.get_figure()
         fig.savefig('img/monthly_doses_germany.png')
```

Vaccination Campaign Progress

```
In [32]: doses_cumulative = vaccinations.loc[ : , ['date', 'personen_erst_kumulativ', 'personen_voll_kumulativ']]
         doses_cumulative.set_index('date', inplace=True)
         doses_cumulative.tail(3)
```

Out[32]:

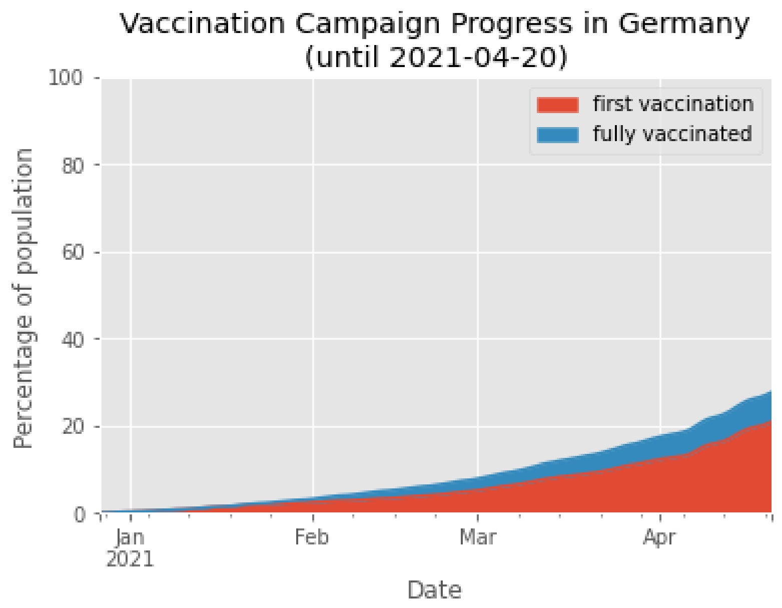
	personen_erst_kumulativ	personen_voll_kumulativ
date		
2021-04-18	16484590	5530428
2021-04-19	16811484	5588910
2021-04-20	17288804	5646788

In [33]:
```
population_germany = 83_200_000
# Calculate new fields
doses_cumulative['first vaccination'] = round(
    doses_cumulative['personen_erst_kumulativ'] * 100 / population_germany,
    2)
doses_cumulative['fully vaccinated'] = round(
    doses_cumulative['personen_voll_kumulativ'] * 100 / population_germany,
    2)
doses_cumulative.drop(columns=['personen_erst_kumulativ','personen_voll_kumulativ'], inplace=True)
doses_cumulative.tail(3)
```

Out[33]:

	first vaccination	fully vaccinated
date		
2021-04-18	19.81	6.65
2021-04-19	20.21	6.72
2021-04-20	20.78	6.79

In [34]:
```
doses_area_plot = doses_cumulative.plot.area(
    ylim=(0,100),
    xlabel='Date',
    ylabel='Percentage of population',
    title=f"Vaccination Campaign Progress in Germany\n(until {last_update})")
```



```
In [35]: fig = doses_area_plot.get_figure()
         fig.savefig('img/vaccinations_germany_area_plot.png')
```

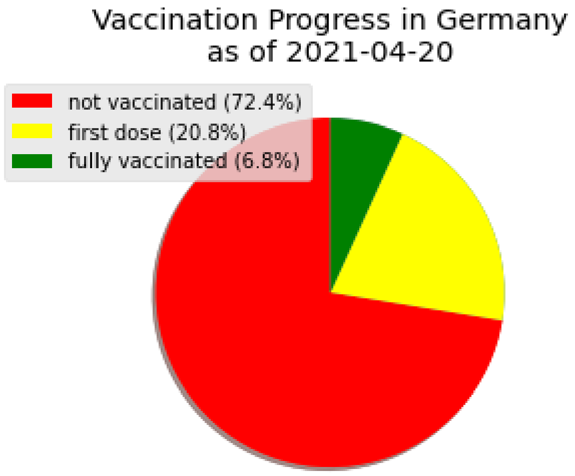
As of Today

```
In [36]: # get the last line of the data
         current_state = doses_cumulative.iloc[-1]
         current_state
```

```
Out[36]: first vaccination    20.78
         fully vaccinated      6.79
         Name: 2021-04-20 00:00:00, dtype: float64
```

```
In [37]: percentage_not_vacc = 100 - current_state['first vaccination'] - current_state['fully vaccinated']
         labels = [f"not vaccinated ({round(percentage_not_vacc, 1)}%)",
                   f"first dose ({round(current_state['first vaccination'],1)}%)",
                   f"fully vaccinated ({round(current_state['fully vaccinated'],1)}%)"]
         colors = ['red', 'yellow', 'green']
         sizes = [percentage_not_vacc,
                  current_state['first vaccination'],
                  current_state['fully vaccinated']]
         fig1, ax1 = plt.subplots()
         ax1.pie(sizes, shadow=True, startangle=90)
```

```
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
patches, texts = plt.pie(sizes, colors=colors, startangle=90)
plt.legend(patches, labels, loc="best")
plt.title(f"Vaccination Progress in Germany\nas of {last_update}")
plt.show()
```



Vaccines in Use

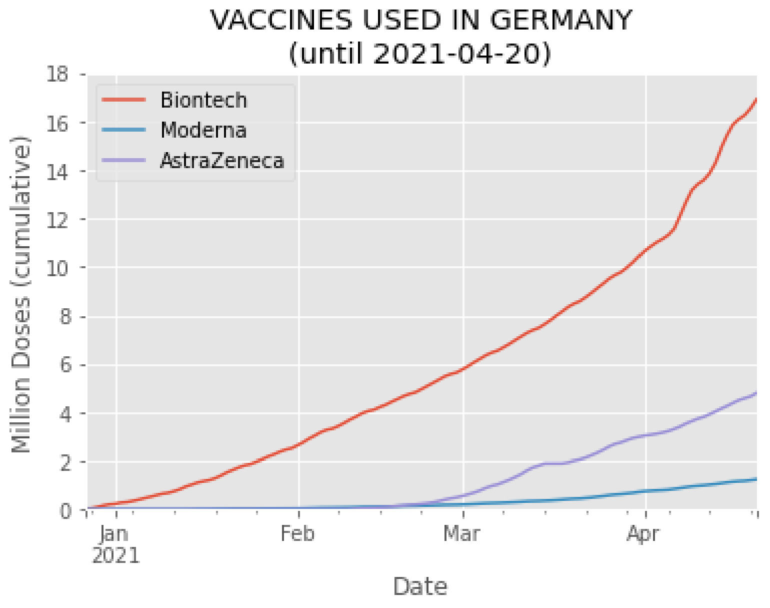
In [38]:
```
vaccine_use = vaccinations.loc[ : , ['date', 'dosen_biontech_kumulativ',
                                     'dosen_moderna_kumulativ',
                                     'dosen_astrazeneca_kumulativ']]
# Rename columns
vaccine_use.columns = ['date', 'Biontech', 'Moderna', 'AstraZeneca']
# make 'date' an index
vaccine_use.set_index('date', inplace=True)
# divide columns by 1 million
vaccine_use["Biontech"] = vaccine_use["Biontech"] / 1_000_000
vaccine_use["Moderna"] = vaccine_use["Moderna"] / 1_000_000
vaccine_use["AstraZeneca"] = vaccine_use["AstraZeneca"] / 1_000_000
vaccine_use.tail(3)
```

Out[38]:

	Biontech	Moderna	AstraZeneca
date			
2021-04-18	16.261274	1.178493	4.575251

	Biontech	Moderna	AstraZeneca
date			
2021-04-19	16.541372	1.207842	4.651180
2021-04-20	16.905070	1.242388	4.788134

In [39]:
```
vaccines_used = vaccine_use.plot(
    # as it is cumulative, the last row must contain the single highest number
    ylim=(0,math.ceil(max(vaccine_use.iloc[-1]))+1),
    xlabel='Date',
    ylabel='Million Doses (cumulative)',
    title=f"VACCINES USED IN GERMANY\n(until {last_update})")
```



In [40]:
```
fig = vaccines_used.get_figure()
fig.savefig('img/vaccines_used_in_germany.png')
```

In [ ]: