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 [49]:
```
# standard library
import datetime
import math
```

In [50]:
```
# third party
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
```

Date this Notebook was run

In [51]:
```
today = datetime.datetime.today().strftime('%Y-%m-%d')
today
```

Out[51]:
```
'2021-04-22'
```

Set Defaults

In [52]:
```
# style like ggplot in R
plt.style.use('ggplot')
```

In [53]:
```
# Avoid cutting off part of the axis labels, see:
# https://stackoverflow.com/questions/6774086/why-is-my-xlabel-cut-off-in-my-matplotlib-plot
plt.rcParams.update({'figure.autolayout': True})
```

Get and Transform Data

In [54]:
```
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 [55]:
```
# No analysis of indication planned:
cols_to_drop = vaccinations.columns[vaccinations.columns.str.contains('indikation_')]
vaccinations.drop(columns=cols_to_drop, inplace=True)
```

In [56]:
```
# Convert datatype of date column
vaccinations.iloc[ : , [0]] = vaccinations.iloc[ : , [0]].apply(pd.to_datetime)
```

Show Data

In [57]:
```
vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 116 entries, 0 to 115
Data columns (total 12 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   date                             116 non-null    datetime64[ns]
 1   dosen_kumulativ                  116 non-null    int64
 2   dosen_differenz_zum_vortag       116 non-null    int64
 3   dosen_erst_differenz_zum_vortag  116 non-null    int64
 4   dosen_zweit_differenz_zum_vortag 116 non-null    int64
 5   dosen_biontech_kumulativ         116 non-null    int64
 6   dosen_moderna_kumulativ          116 non-null    int64
 7   dosen_astrazeneca_kumulativ      116 non-null    int64
 8   personen_erst_kumulativ          116 non-null    int64
 9   personen_voll_kumulativ          116 non-null    int64
 10  impf_quote_erst                  116 non-null    float64
 11  impf_quote_voll                  116 non-null    float64
dtypes: datetime64[ns](1), float64(2), int64(9)
memory usage: 11.0 KB
```

In [58]:
```
vaccinations.tail(3)
```

Out[58]:

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
113	2021-04-19	22427351	393222	333944	59278	1655733
114	2021-04-20	22967899	540548	481688	58860	1692512
115	2021-04-21	23656941	689042	621755	67287	1735936

Last Update

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

In [59]:
```
last_update = vaccinations.loc[vaccinations.index[-1], "date"].strftime('%Y-%m-%d')
last_update
```

Out[59]:
```
'2021-04-21'
```

Doses Used

In [60]:
```
doses = vaccinations.loc[ : , ['date', 'dosen_differenz_zum_vortag']]
# Rename columns
doses.columns = ['date', 'doses used']
```

In [61]:
```
# Scale number of doses as millions
doses['doses used'] = doses['doses used'] / 1_000_000
```

Doses Daily

In [62]:
```
doses_daily = doses.set_index('date', inplace=False)
doses_daily.tail(1)
```

Out[62]:

	doses used
date	

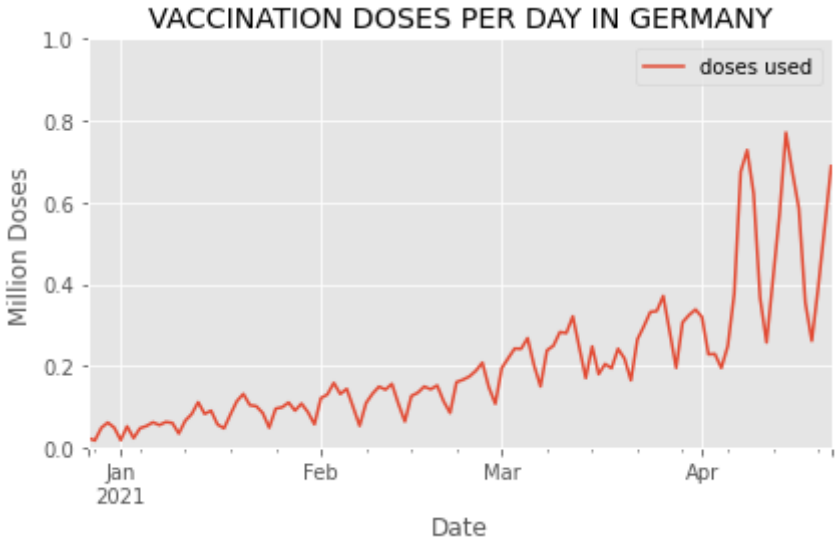
date	doses used
2021-04-21	0.689042

In [63]:
```
# What is the highest number of doses used in a day?
max_doses_daily = max(doses_daily['doses used'])
max_doses_daily
```

Out[63]: 0.771243

In [64]:
```
doses_daily.plot(
    ylim=(0,math.ceil(max_doses_daily)),
    xlabel='Date',
    ylabel='Million Doses',
    title='VACCINATION DOSES PER DAY IN GERMANY')
```

Out[64]: <AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>



Doses per Weekday (in the last 6 weeks)

In [65]:
```
last_6_weeks = doses.tail(42)
```

In [66]:
```
# 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-66-45013977109e>: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 [67]:
```
# check:
last_6_weeks.tail(3)
```

Out[67]:

	date	doses used	weekday
113	2021-04-19	0.393222	Monday
114	2021-04-20	0.540548	Tuesday
115	2021-04-21	0.689042	Wednesday

In [68]:
```
# drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

In [69]:
```
#last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

Out[69]:

	doses used	weekday
113	0.393222	Monday
114	0.540548	Tuesday
115	0.689042	Wednesday

In [70]:
```
pivot_table =last_6_weeks.pivot(columns='weekday', values='doses used')
pivot_table.tail()
```

Out[70]:

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
111	NaN	NaN	0.355843	NaN	NaN	NaN	NaN
112	NaN	NaN	NaN	0.262002	NaN	NaN	NaN

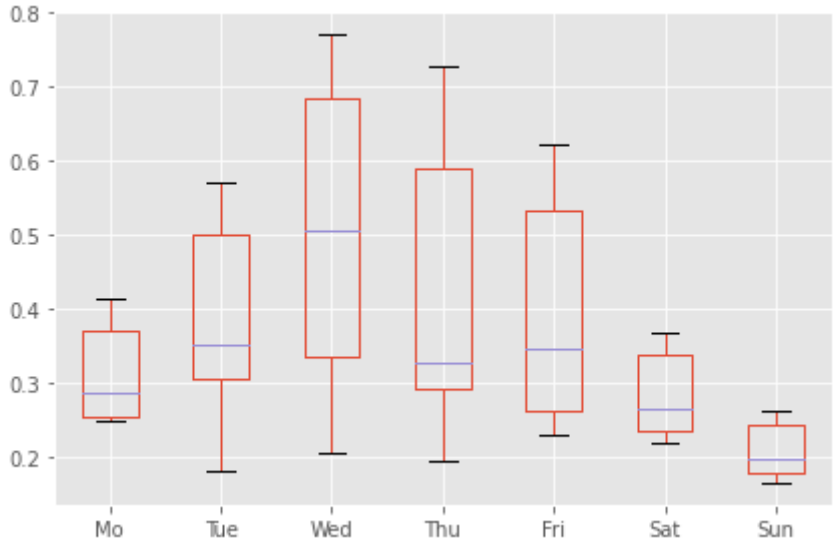
weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
113	NaN	0.393222	NaN	NaN	NaN	NaN	NaN
114	NaN	NaN	NaN	NaN	NaN	0.540548	NaN
115	NaN	NaN	NaN	NaN	NaN	NaN	0.689042

In [71]:
```
# 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[71]:

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
111	NaN	NaN	NaN	NaN	NaN	0.355843	NaN
112	NaN	NaN	NaN	NaN	NaN	NaN	0.262002
113	0.393222	NaN	NaN	NaN	NaN	NaN	NaN
114	NaN	0.540548	NaN	NaN	NaN	NaN	NaN
115	NaN	NaN	0.689042	NaN	NaN	NaN	NaN

In [72]:
```
weekday_boxplot = pivot_table.boxplot()
```



In [73]:
```
fig = weekday_boxplot.get_figure()
fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

In [74]:
```
# 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[74]:

date	million doses used
2021-03-29	2.124394
2021-04-05	1.888266
2021-04-12	3.442376
2021-04-19	3.612712
2021-04-26	1.229590

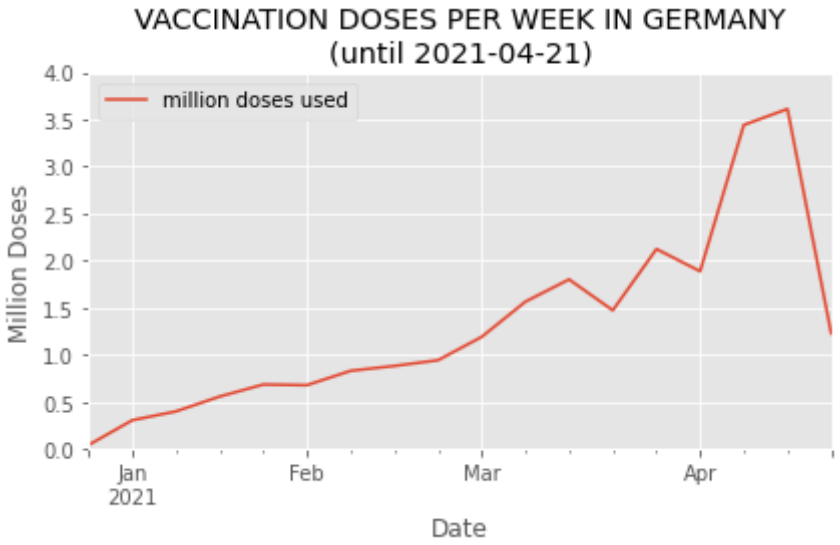
In [75]:
```
# What is the highest number of doses used in a week?
```

```
max_million_doses_weekly = max(doses_weekly['million doses used'])
max_million_doses_weekly
```

Out[75]: 3.612712

In [76]:
```
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[76]: <AxesSubplot:title={'center':'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-04-21)'}, xlabel='Date', ylabel='Million Doses'>



Doses per Month

In [77]:
```
# M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date',freq='M')).sum()
doses_monthly.tail()
```

Out[77]:

	doses used
date	
2020-12-31	0.204160

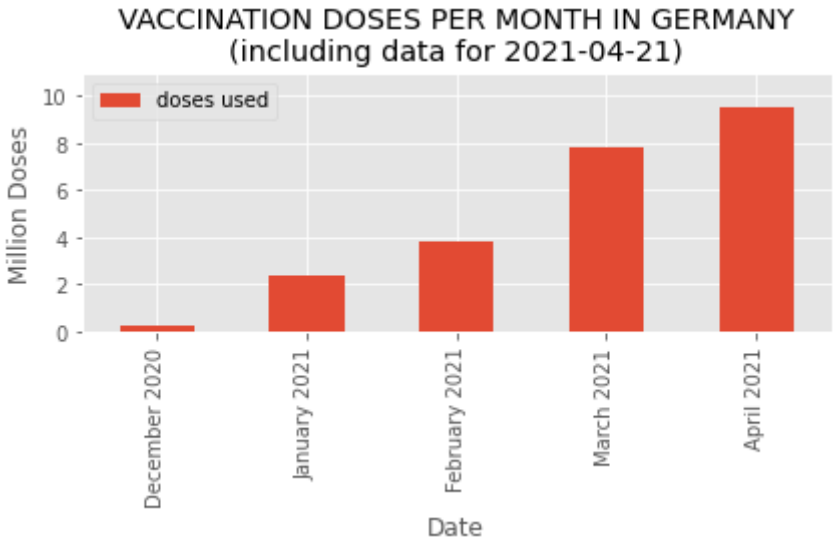
date	doses used
2021-01-31	2.344017
2021-02-28	3.775305
2021-03-31	7.824093
2021-04-30	9.509366

In [78]:
```
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[78]:

label	doses used
December 2020	0.204160
January 2021	2.344017
February 2021	3.775305
March 2021	7.824093
April 2021	9.509366

In [79]:
```
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 [80]:
```
fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_germany.png')
```

Vaccination Campaign Progress

In [81]:
```
doses_cumulative = vaccinations.loc[ : , ['date', 'personen_erst_kumulativ', 'personen_voll_kumulativ']]
doses_cumulative.set_index('date', inplace=True)
doses_cumulative.tail(3)
```

Out[81]:

date	personen_erst_kumulativ	personen_voll_kumulativ
2021-04-19	16828937	5598414
2021-04-20	17310625	5657274
2021-04-21	17932380	5724561

In [82]:
```
population_germany = 83_200_000
# Calculate new fields
doses_cumulative['first vaccination'] = round(
    doses_cumulative['personen_erst_kumulativ'] * 100 / population_germany,
    2)
```

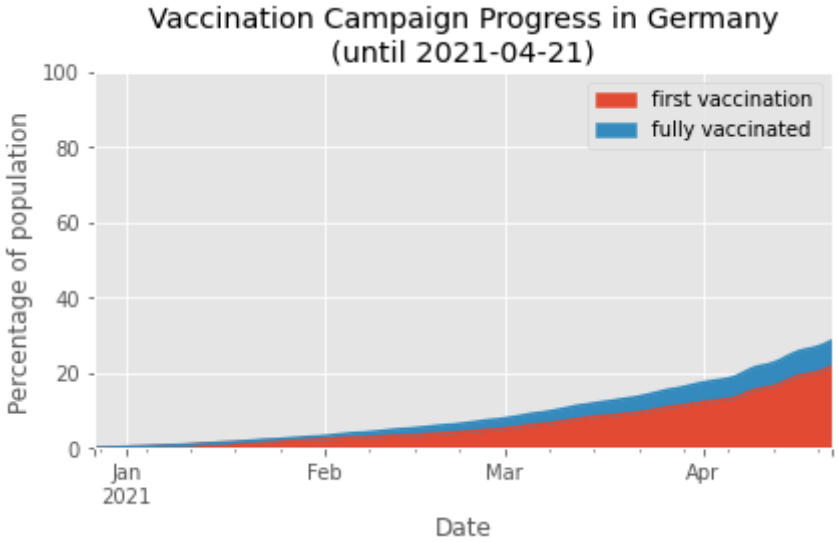
```
doses_cumulative['fully vaccinated'] = round(
    doses_cumulative['personen_voll_kumulativ'] * 100 / population_germany,
    2)
doses_cumulative.drop(columns=['personen_erst_kumulativ','personen_voll_kumulativ'], inplace=True)
doses_cumulative.tail(3)
```

Out[82]:

date	first vaccination	fully vaccinated
2021-04-19	20.23	6.73
2021-04-20	20.81	6.80
2021-04-21	21.55	6.88

In [83]:

```
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 [84]:

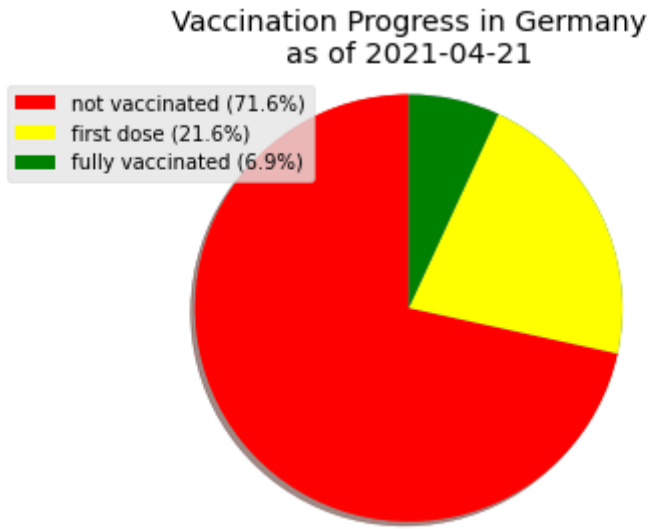
```
fig = doses_area_plot.get_figure()
fig.savefig('img/vaccinations_germany_area_plot.png')
```

As of Today

In [85]:
```
# get the last line of the data
current_state = doses_cumulative.iloc[-1]
current_state
```

Out[85]:
```
first vaccination    21.55
fully vaccinated      6.88
Name: 2021-04-21 00:00:00, dtype: float64
```

In [86]:
```
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.savefig must be before show()
# BEWARE plt.savefig must be in the same Jupyter code cell that creates the graph!
# See comment by ijoseph here:
# https://stackoverflow.com/questions/9012487/matplotlib-pyplot-savefig-outputs-blank-image
plt.savefig('img/vaccination_in_germany_pie.png', bbox_inches='tight')
plt.show()
```



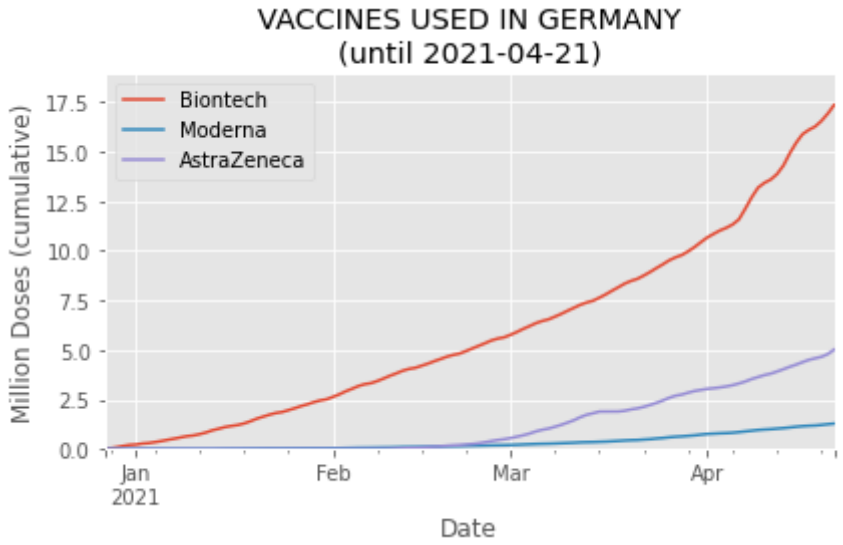
Vaccines in Use

In [87]:
```
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[87]:

date	Biontech	Moderna	AstraZeneca
2021-04-19	16.557335	1.216213	4.653803
2021-04-20	16.925126	1.251506	4.791267
2021-04-21	17.359368	1.284116	5.013457

In [88]:
```
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 [89]:
```
fig = vaccines_used.get_figure()
fig.savefig('img/vaccines_used_in_germany.png')
```

In [ ]: