

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]: '2021-04-29'
```

Set Defaults

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

```
In [5]: # 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 [6]: 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 [7]: # No analysis of indication planned:
cols_to_drop = vaccinations.columns[vaccinations.columns.str.contains('indikation_')]
vaccinations.drop(columns=cols_to_drop, inplace=True)
```

```
In [8]: # Convert datatype of date column
vaccinations.iloc[ : , [0]] = vaccinations.iloc[ : , [0]].apply(pd.to_datetime)
```

Show Data

```
In [9]: vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123 entries, 0 to 122
Data columns (total 21 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   date                                     123 non-null    datetime64[ns]
1   dosen_kumulativ                         123 non-null    int64
2   dosen_differenz_zum_vortag              123 non-null    int64
3   dosen_erst_differenz_zum_vortag          123 non-null    int64
4   dosen_zweit_differenz_zum_vortag         123 non-null    int64
5   dosen_biontech_kumulativ                123 non-null    int64
6   dosen_moderna_kumulativ                 123 non-null    int64
7   dosen_astrazeneca_kumulativ              123 non-null    int64
8   personen_erst_kumulativ                  123 non-null    int64
9   personen_voll_kumulativ                  123 non-null    int64
10  impf_quote_erst                          123 non-null    float64
11  impf_quote_voll                          123 non-null    float64
12  dosen_dim_kumulativ                      123 non-null    int64
13  dosen_kbv_kumulativ                      123 non-null    int64
14  dosen_johnson_kumulativ                  123 non-null    int64
```

```

15  dosen_biontech_erst_kumulativ      123 non-null    int64
16  dosen_biontech_zweit_kumulativ     123 non-null    int64
17  dosen_moderna_erst_kumulativ       123 non-null    int64
18  dosen_moderna_zweit_kumulativ      123 non-null    int64
19  dosen_astazeneca_erst_kumulativ    123 non-null    int64
20  dosen_astazeneca_zweit_kumulativ   123 non-null    int64
dtypes: datetime64[ns](1), float64(2), int64(18)
memory usage: 20.3 KB

```

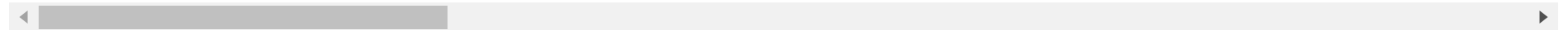
vaccination

```
In [10]: vaccinations.tail(3)
```

```
Out[10]:
```

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulativ
120	2021-04-26	25933969	412839	342081	70758	1882373
121	2021-04-27	26714827	780858	693760	87098	1950472
122	2021-04-28	27803779	1088952	970295	118657	2049411

3 rows × 21 columns



Last Update

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

```
In [11]: last_update = vaccinations.loc[vaccinations.index[-1], "date"].strftime('%Y-%m-%d')
last_update
```

```
Out[11]: '2021-04-28'
```

Doses Used

```
In [12]: doses = vaccinations.loc[:, ['date', 'dosen_differenz_zum_vortag']]
# Rename columns
doses.columns = ['date', 'doses used']
```

```
In [13]: # Scale number of doses as millions
doses['doses used'] = doses['doses used'] / 1_000_000
```

Doses Daily

```
In [14]: doses_daily = doses.set_index('date', inplace=False)
doses_daily.tail(1)
```

```
Out[14]:
```

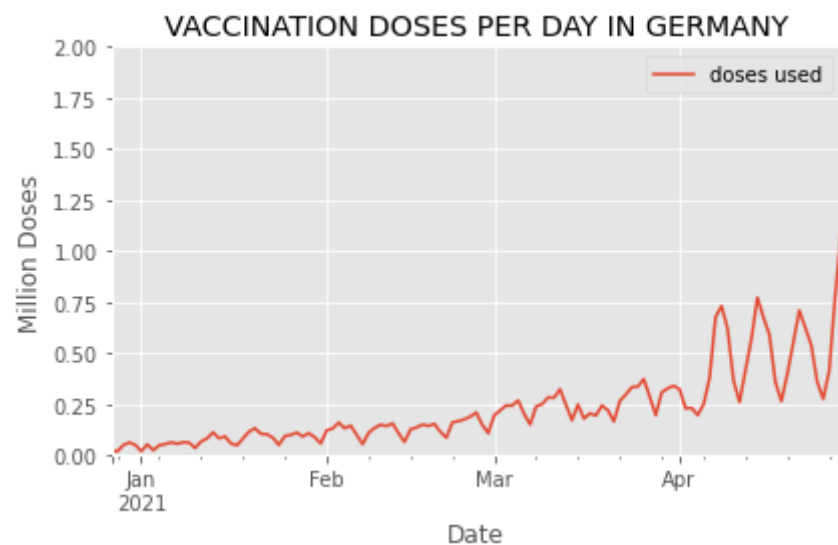
doses used	
date	
2021-04-28	1.088952

```
In [15]: # What is the highest number of doses used in a day?
max_doses_daily = max(doses_daily['doses used'])
max_doses_daily
```

```
Out[15]: 1.088952
```

```
In [16]: doses_daily.plot(
    ylim=(0,math.ceil(max_doses_daily)),
    xlabel='Date',
    ylabel='Million Doses',
    title='VACCINATION DOSES PER DAY IN GERMANY')
```

```
Out[16]: <AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>
```



Doses per Weekday (in the last 6 weeks)

```
In [17]: last_6_weeks = doses.tail(42)
```

```
In [18]: # 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-18-45013977109e>: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 [19]: # check:
last_6_weeks.tail(3)
```

```
Out[19]:
```

	date	doses used	weekday
120	2021-04-26	0.412839	Monday
121	2021-04-27	0.780858	Tuesday
122	2021-04-28	1.088952	Wednesday

```
In [20]: # drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

```
In [21]: #last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

```
Out[21]:
```

	doses used	weekday
120	0.412839	Monday
121	0.780858	Tuesday
122	1.088952	Wednesday

```
In [22]: pivot_table = last_6_weeks.pivot(columns='weekday', values='doses used')
```

```
pivot_table.tail()
```

```
Out[22]:
```

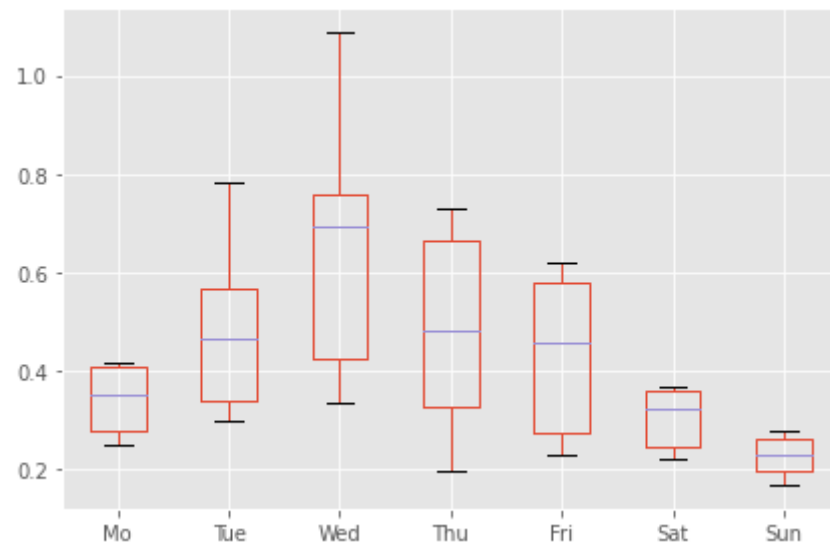
	weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
118		NaN	NaN	0.359128	NaN	NaN	NaN	NaN
119		NaN	NaN	NaN	0.275838	NaN	NaN	NaN
120		NaN	0.412839	NaN	NaN	NaN	NaN	NaN
121		NaN	NaN	NaN	NaN	NaN	0.780858	NaN
122		NaN	NaN	NaN	NaN	NaN	NaN	1.088952

```
In [23]: # 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[23]:
```

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
118	NaN	NaN	NaN	NaN	NaN	0.359128	NaN
119	NaN	NaN	NaN	NaN	NaN	NaN	0.275838
120	0.412839	NaN	NaN	NaN	NaN	NaN	NaN
121	NaN	0.780858	NaN	NaN	NaN	NaN	NaN
122	NaN	NaN	1.088952	NaN	NaN	NaN	NaN

```
In [24]: weekday_boxplot = pivot_table.boxplot()
```



```
In [25]: fig = weekday_boxplot.get_figure()
fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

```
In [26]: # 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[26]:

million doses used	
date	
2021-04-05	1.895936
2021-04-12	3.447962
2021-04-19	3.630889
2021-04-26	3.474820
2021-05-03	1.869810

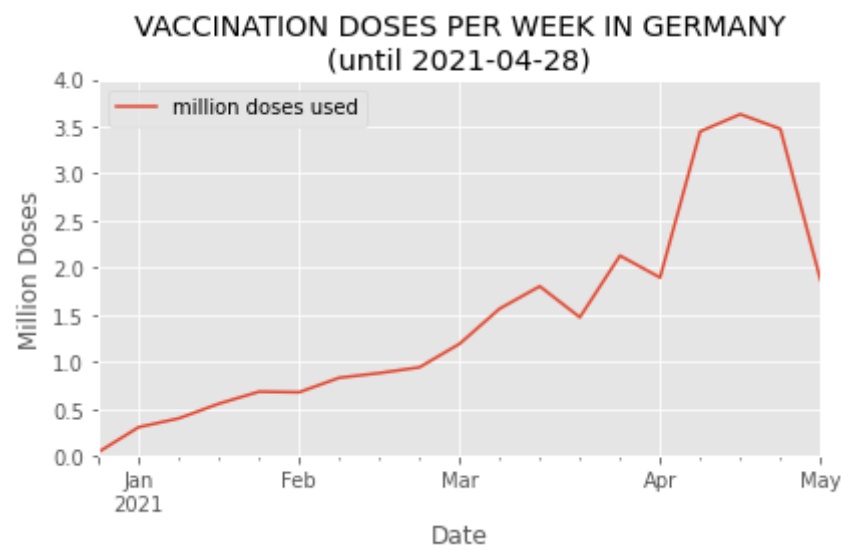
```
In [27]: # What is the highest number of doses used in a week?
```

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

Out[27]: 3.6308890000000003

```
In [28]: 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[28]: <AxesSubplot:title={'center': 'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-04-28)'}, xlabel='Date', ylabel='Million Doses'>



Doses per Month

```
In [29]: # M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date', freq='M')).sum()
doses_monthly.tail()
```

Out[29]:

doses used	
date	
2020-12-31	0.204544

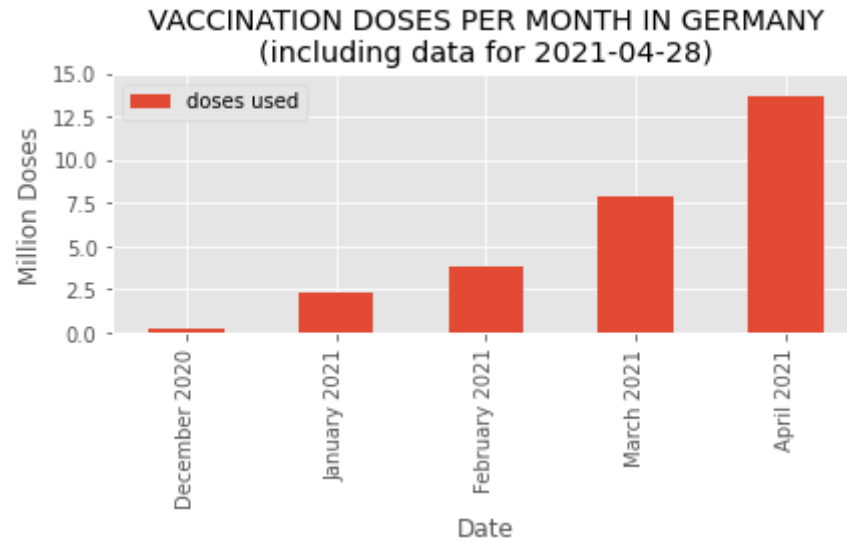
doses used	
date	
2021-01-31	2.340455
2021-02-28	3.773364
2021-03-31	7.833405
2021-04-30	13.652011

```
In [30]: 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[30]:
```

doses used	
label	
December 2020	0.204544
January 2021	2.340455
February 2021	3.773364
March 2021	7.833405
April 2021	13.652011

```
In [31]: 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 [32]: fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_germany.png')
```

Vaccination Campaign Progress

```
In [33]: doses_cumulative = vaccinations.loc[:, ['date', 'personen_erst_kumulativ', 'personen_voll_kumulativ']]
doses_cumulative.set_index('date', inplace=True)
doses_cumulative.tail(3)
```

```
Out[33]:
```

	personen_erst_kumulativ	personen_voll_kumulativ
date		
2021-04-26	19898572	6035397
2021-04-27	20592332	6122495
2021-04-28	21562627	6241152

```
In [34]: 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[34]:

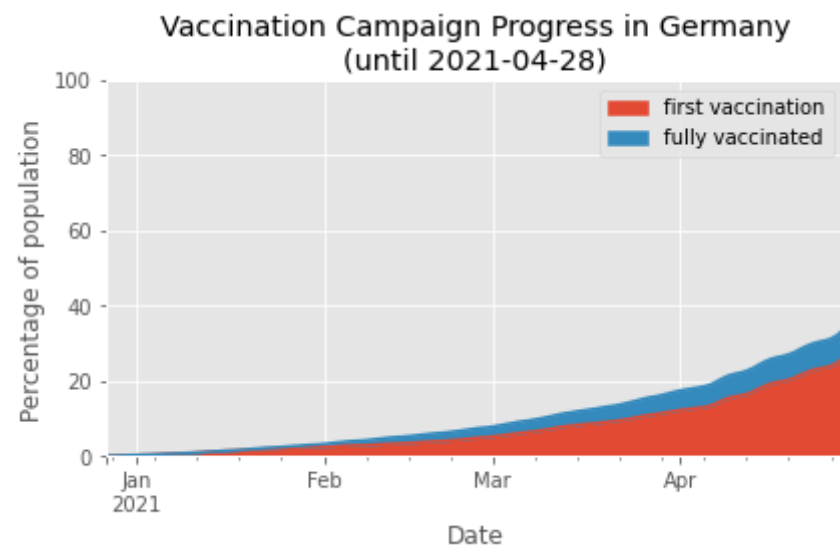
	first vaccination	fully vaccinated
date		

date		
2021-04-26	23.92	7.25
2021-04-27	24.75	7.36
2021-04-28	25.92	7.50

```

In [35]: 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 [36]: fig = doses_area_plot.get_figure()
fig.savefig('img/vaccinations_germany_area_plot.png')

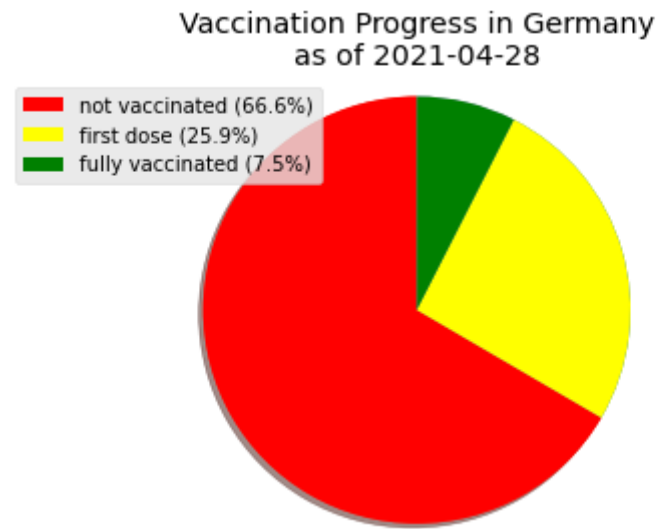
```

As of Today

```
In [37]: # get the last line of the data
current_state = doses_cumulative.iloc[-1]
current_state
```

```
Out[37]: first vaccination    25.92
fully vaccinated           7.50
Name: 2021-04-28 00:00:00, dtype: float64
```

```
In [38]: 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 ioseph 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

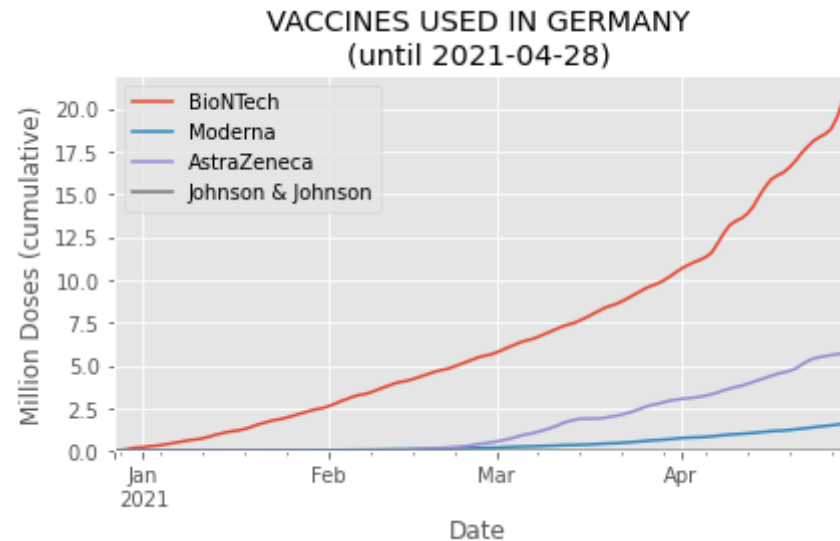
```
In [39]: vaccine_use = vaccinations.loc[ : , ['date', 'dosen_biontech_kumulativ',
                                             'dosen_moderna_kumulativ',
                                             'dosen_astrazeneca_kumulativ',
                                             'dosen_johnson_kumulativ']]

# Rename columns
vaccine_use.columns = ['date', 'BioNTech', 'Moderna', 'AstraZeneca', 'Johnson & Johnson']
# 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["Johnson & Johnson"] = vaccine_use["Johnson & Johnson"] / 1_000_000
vaccine_use.tail(3)
```

```
Out[39]:
```

	BioNTech	Moderna	AstraZeneca	Johnson & Johnson
date				
2021-04-26	18.823733	1.499469	5.610705	0.000062
2021-04-27	19.504721	1.544639	5.665016	0.000451
2021-04-28	20.494118	1.592535	5.716115	0.001011

```
In [40]: 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 [41]: fig = vaccines_used.get_figure()  
fig.savefig('img/vaccines_used_in_germany.png')
```