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 [50]:
```
# standard library
import datetime
import math
```

In [51]:
```
# third party
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
```

Date this Notebook was run

In [52]:
```
today = datetime.datetime.today().strftime('%Y-%m-%d')
today
```

Out[52]:
```
'2021-04-27'
```

Set Defaults

In [53]:
```
# style like ggplot in R
plt.style.use('ggplot')
```

In [54]:
```
# 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 [55]:
```
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 [56]:
```
# No analysis of indication planned:
cols_to_drop = vaccinations.columns[vaccinations.columns.str.contains('indikation_')]
vaccinations.drop(columns=cols_to_drop, inplace=True)
```

In [57]:
```
# Convert datatype of date column
vaccinations.iloc[ : , [0]] = vaccinations.iloc[ : , [0]].apply(pd.to_datetime)
```

Show Data

In [58]:
```
vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121 entries, 0 to 120
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   date                            121 non-null    datetime64[ns]
 1   dosen_kumulativ                 121 non-null    int64
 2   dosen_differenz_zum_vortag      121 non-null    int64
 3   dosen_erst_differenz_zum_vortag 121 non-null    int64
 4   dosen_zweit_differenz_zum_vortag 121 non-null   int64
 5   dosen_biontech_kumulativ        121 non-null    int64
 6   dosen_moderna_kumulativ         121 non-null    int64
 7   dosen_astrazeneca_kumulativ     121 non-null    int64
 8   personen_erst_kumulativ         121 non-null    int64
 9   personen_voll_kumulativ         121 non-null    int64
 10  impf_quote_erst                 121 non-null    float64
 11  impf_quote_voll                 121 non-null    float64
 12  dosen_dim_kumulativ             121 non-null    int64
 13  dosen_kbv_kumulativ             121 non-null    int64
 14  dosen_johnson_kumulativ         121 non-null    int64
```

```
dtypes: datetime64[ns](1), float64(2), int64(12)
memory usage: 14.3 KB
```

In [59]:
```
vaccinations.tail(3)
```

Out[59]:

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
118	2021-04-24	25219240	350996	294813	56183	183444
119	2021-04-25	25489913	270673	223953	46720	185276
120	2021-04-26	25892613	402700	333009	69691	1883041

Last Update

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

In [60]:
```
last_update = vaccinations.loc[vaccinations.index[-1], "date"].strftime('%Y-%m-%d')
last_update
```

Out[60]:
```
'2021-04-26'
```

Doses Used

In [61]:
```
doses = vaccinations.loc[ : , ['date', 'dosen_differenz_zum_vortag']]
# Rename columns
doses.columns = ['date', 'doses used']
```

In [62]:
```
# Scale number of doses as millions
doses['doses used'] = doses['doses used'] / 1_000_000
```

Doses Daily

In [63]:
```
doses_daily = doses.set_index('date', inplace=False)
doses_daily.tail(1)
```

Out[63]:

doses used

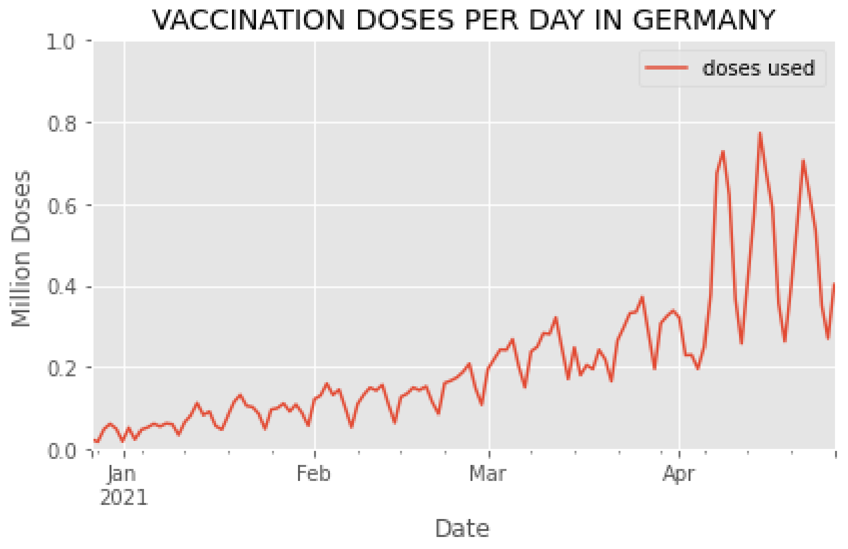
	doses used
date	
2021-04-26	0.4027

In [64]:
```
# What is the highest number of doses used in a day?
max_doses_daily = max(doses_daily['doses used'])
max_doses_daily
```

Out[64]:
```
0.772655
```

In [65]:
```
doses_daily.plot(
    ylim=(0,math.ceil(max_doses_daily)),
    xlabel='Date',
    ylabel='Million Doses',
    title='VACCINATION DOSES PER DAY IN GERMANY')
```

Out[65]:
```
<AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>
```



Doses per Weekday (in the last 6 weeks)

In [66]:
```
last_6_weeks = doses.tail(42)
```

In [67]:
```
# 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-67-45013977109e>: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 [68]:
```
# check:
last_6_weeks.tail(3)
```

Out[68]:

	date	doses used	weekday
118	2021-04-24	0.350996	Saturday
119	2021-04-25	0.270673	Sunday
120	2021-04-26	0.402700	Monday

In [69]:
```
# drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

In [70]:
```
#last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

Out[70]:

	doses used	weekday
118	0.350996	Saturday
119	0.270673	Sunday
120	0.402700	Monday

In [71]:
```
pivot_table =last_6_weeks.pivot(columns='weekday', values='doses used')
pivot_table.tail()
```

Out[71]:

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
116	NaN	NaN	NaN	NaN	0.622781	NaN	NaN
117	0.534503	NaN	NaN	NaN	NaN	NaN	NaN

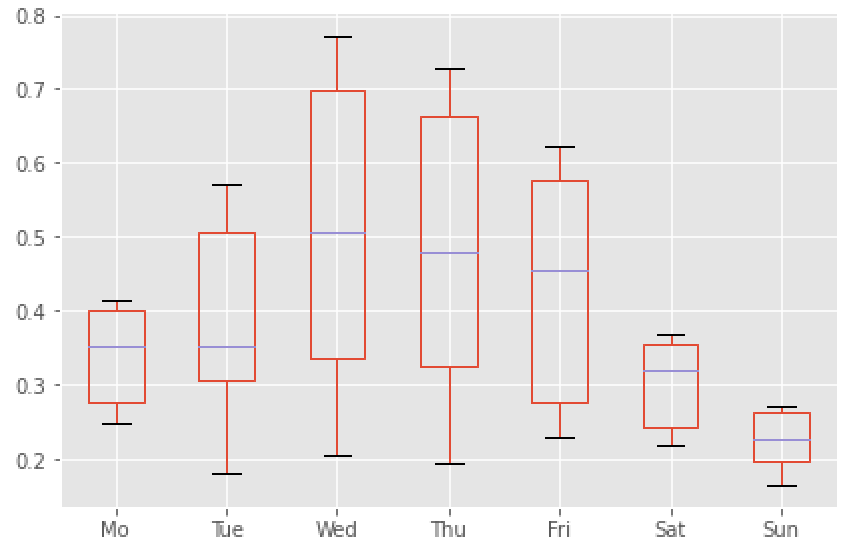
weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
118	NaN	NaN	0.350996	NaN	NaN	NaN	NaN
119	NaN	NaN	NaN	0.270673	NaN	NaN	NaN
120	NaN	0.4027	NaN	NaN	NaN	NaN	NaN

In [72]:
```
# 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[72]:

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
116	NaN	NaN	NaN	0.622781	NaN	NaN	NaN
117	NaN	NaN	NaN	NaN	0.534503	NaN	NaN
118	NaN	NaN	NaN	NaN	NaN	0.350996	NaN
119	NaN	NaN	NaN	NaN	NaN	NaN	0.270673
120	0.4027	NaN	NaN	NaN	NaN	NaN	NaN

In [73]:
```
weekday_boxplot = pivot_table.boxplot()
```



In [74]:
```
fig = weekday_boxplot.get_figure()
fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

In [75]:
```
# 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[75]:

	million doses used
date	
2021-03-29	2.126714
2021-04-05	1.891301
2021-04-12	3.444905
2021-04-19	3.623889
2021-04-26	3.437558

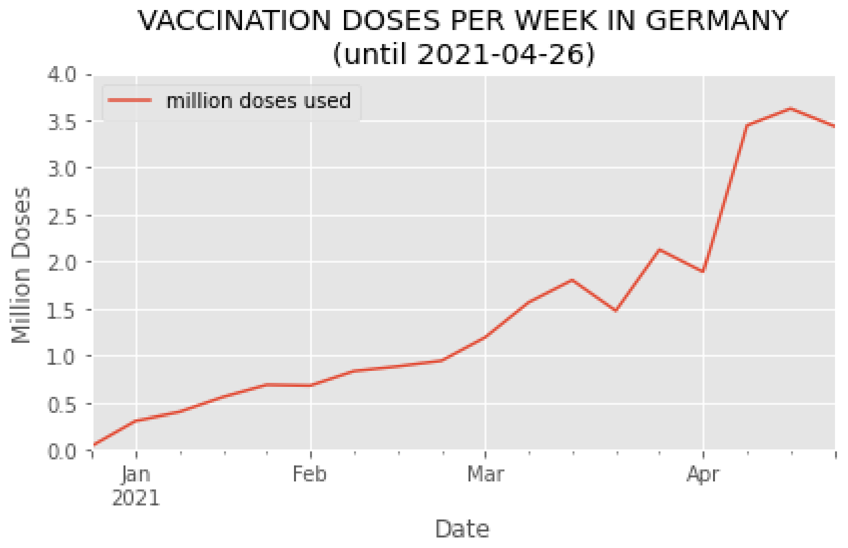
In [76]:
```
# What is the highest number of doses used in a week?
```

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

Out[76]:
```
3.6238889999999997
```

In [77]:
```
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[77]:
```
<AxesSubplot:title={'center':'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-04-26)'}, xlabel='Date', ylabel='Milli
on Doses'>
```



Doses per Month

In [78]:
```
# M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date',freq='M')).sum()
doses_monthly.tail()
```

Out[78]:

	doses used
date	
2020-12-31	0.204883

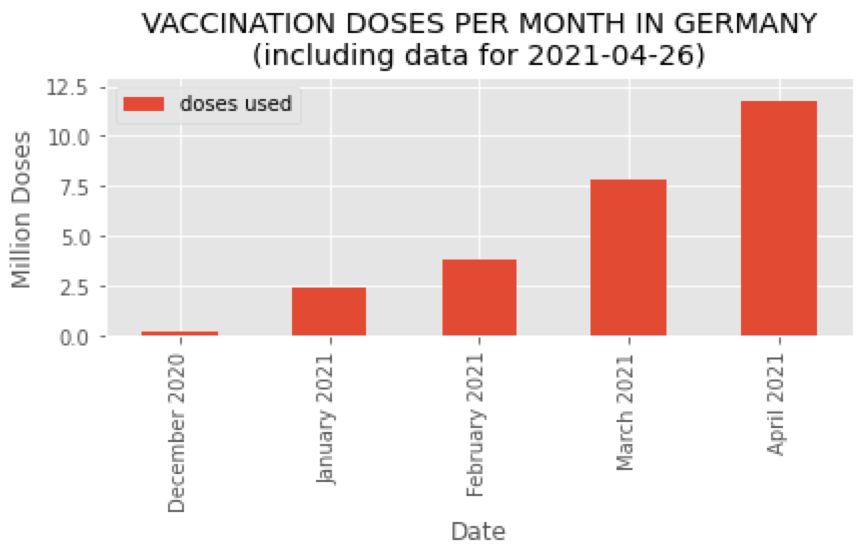
	doses used
date	
2021-01-31	2.346848
2021-02-28	3.777572
2021-03-31	7.829747
2021-04-30	11.733563

In [79]:
```
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[79]:

	doses used
label	
December 2020	0.204883
January 2021	2.346848
February 2021	3.777572
March 2021	7.829747
April 2021	11.733563

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

Vaccination Campaign Progress

```
In [82]: doses_cumulative = vaccinations.loc[ : , ['date', 'personen_erst_kumulativ', 'personen_voll_kumulativ']]
         doses_cumulative.set_index('date', inplace=True)
         doses_cumulative.tail(3)
```

Out[82]:

	personen_erst_kumulativ	personen_voll_kumulativ
date		
2021-04-24	19297588	5921652
2021-04-25	19521541	5968372
2021-04-26	19854550	6038063

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

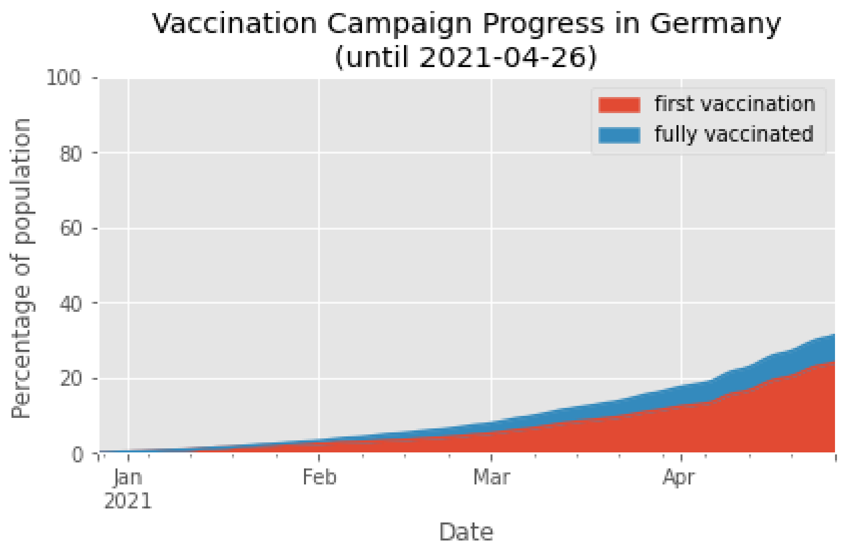
```
doses_cumulative['fully vaccinated'] = round(
    doses_cumulative['personen_voll_kumulativ'] * 100 / population_germany,
    2)
doses_cumulative.drop(columns=['personen_erst_kumulativ','personen_voll_kumulativ'], inplace=True)
doses_cumulative.tail(3)
```

Out[83]:

	first vaccination	fully vaccinated
date		
2021-04-24	23.19	7.12
2021-04-25	23.46	7.17
2021-04-26	23.86	7.26

In [84]:

```
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 [85]:

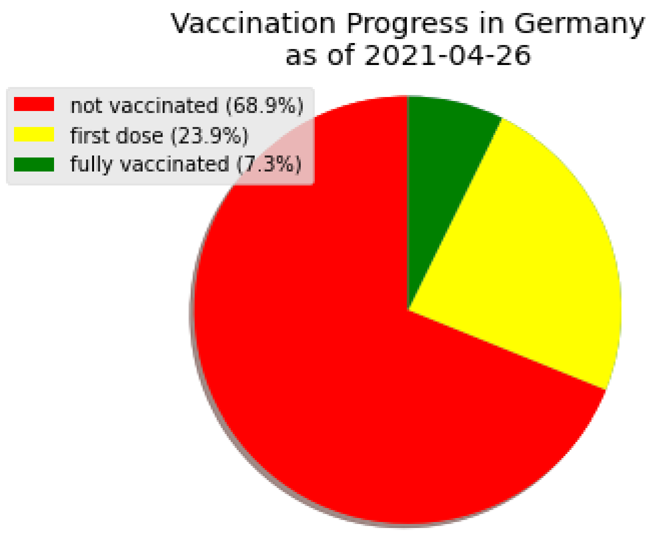
```
fig = doses_area_plot.get_figure()
fig.savefig('img/vaccinations_germany_area_plot.png')
```

As of Today

In [86]:
```
# get the last line of the data
current_state = doses_cumulative.iloc[-1]
current_state
```

Out[86]:
```
first vaccination    23.86
fully vaccinated      7.26
Name: 2021-04-26 00:00:00, dtype: float64
```

In [87]:
```
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()
```



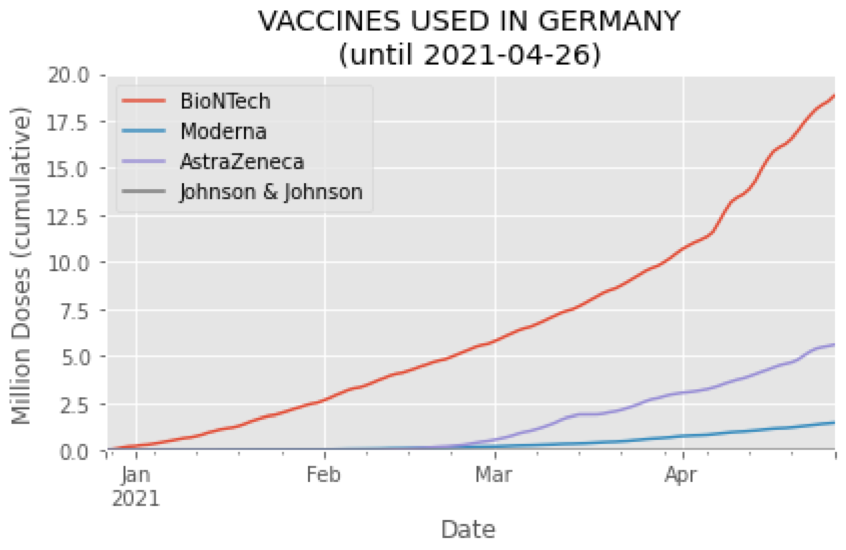
Vaccines in Use

```
In [88]: 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[88]:

date	BioNTech	Moderna	AstraZeneca	Johnson & Johnson
2021-04-24	18.344481	1.408392	5.466367	0.000000
2021-04-25	18.527686	1.435608	5.526619	0.000000
2021-04-26	18.830416	1.466856	5.595278	0.000063

```
In [89]: 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 [90]: fig = vaccines_used.get_figure()
         fig.savefig('img/vaccines_used_in_germany.png')
```