

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

```
In [94]: # third party  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import requests  
import seaborn
```

Date this Notebook was run

```
In [95]: today = datetime.datetime.today().strftime('%Y-%m-%d')  
today
```

```
Out[95]: '2021-05-09'
```

Set Defaults

```
In [96]: # style like ggplot in R  
plt.style.use('ggplot')
```

```
In [97]: # 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})
```

```
In [98]: population_germany = 83_200_000
```

Get and Transform Data

```
In [99]: vaccination_data_permalink = 'https://impfdashboard.de/static/data/germany_vaccinations_timeseries_v2.tsv'
vaccinations = pd.read_csv(
    vaccination_data_permalink,
    sep="\t")
```

Drop unnecessary / misleading columns

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

```
In [100]: cols_to_drop = vaccinations.columns[vaccinations.columns.str.contains('indikation_')]
vaccinations.drop(columns=cols_to_drop, inplace=True)
```

Some more columns can be dropped, as there is no interest in analyzing differences on a vaccine level - especially since in some cases vaccines were mixed.

```
In [101]: more_cols_to_drop = ['dosen_biontech_erst_kumulativ', 'dosen_biontech_zweit_kumulativ',
                             'dosen_moderna_erst_kumulativ', 'dosen_moderna_zweit_kumulativ',
                             'dosen_astrazeneca_erst_kumulativ', 'dosen_astrazeneca_zweit_kumulativ']
vaccinations.drop(columns=more_cols_to_drop, inplace=True)
```

Some columns are labeled misleadingly. As stated by the data provider the columns `personen_erst_kumulativ` and `impf_quote_erst` contain people vaccinated with the Johnson & Johnson vaccine. As this requires only one shot. the same persons are included in `personen_voll_kumulativ`. Therefore more columns are dropped and recalculated later.

```
In [102]: vaccinations.drop(columns=['impf_quote_erst', 'impf_quote_voll'], inplace=True)
```

Convert datatype of date column

```
In [103]: vaccinations.iloc[:, [0]] = vaccinations.iloc[:, [0]].apply(pd.to_datetime)
```

Show Data

```
In [104]: vaccinations.info()

<class 'pandas.core.frame.DataFrame'>
```

RangeIndex: 132 entries, 0 to 131

Data columns (total 13 columns):

#	Column	Non-Null Count	Dtype
0	date	132 non-null	datetime64[ns]
1	dosen_kumulativ	132 non-null	int64
2	dosen_differenz_zum_vortag	132 non-null	int64
3	dosen_erst_differenz_zum_vortag	132 non-null	int64
4	dosen_zweit_differenz_zum_vortag	132 non-null	int64
5	dosen_biontech_kumulativ	132 non-null	int64
6	dosen_moderna_kumulativ	132 non-null	int64
7	dosen_astrazeneca_kumulativ	132 non-null	int64
8	personen_erst_kumulativ	132 non-null	int64
9	personen_voll_kumulativ	132 non-null	int64
10	dosen_dim_kumulativ	132 non-null	int64
11	dosen_kbv_kumulativ	132 non-null	int64
12	dosen_johnson_kumulativ	132 non-null	int64

dtypes: datetime64[ns](1), int64(12)

memory usage: 13.5 KB

In [105... vaccinations.tail(3)

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulativ
129	2021-05-05	32652324	1115886	898480	217406	2427062
130	2021-05-06	33594803	942479	737564	204915	2493674
131	2021-05-07	34408840	814037	617514	196523	2548032

Check Validity

In [106... *# get the last row / the newest available data*
last_row = vaccinations.tail(1)

In [107... doses_used = last_row['dosen_kumulativ']
doses_used

Out[107... 131 34408840
Name: dosen_kumulativ, dtype: int64

```
In [108... # The number of person having been vaccinated at least once, includes those fully vaccinated
at_least_once = last_row['personen_erst_kumulativ']
fully_vaccinated_people = last_row['personen_voll_kumulativ']
partially_vaccinated_people = at_least_once - fully_vaccinated_people
# The johnson & Johnson vaccine is the only one used in Germany that only needs a single shot:
johnson_doses = last_row['dosen_johnson_kumulativ']
```

```
In [109... # Must be exactly 0
doses_used - partially_vaccinated_people - (fully_vaccinated_people - johnson_doses) * 2 - johnson_doses == 0
```

```
Out[109... 131      True
dtype: bool
```

Calculate columns

```
In [110... vaccinations['partly vaccinated'] = round(
    (vaccinations['personen_erst_kumulativ'] - vaccinations['personen_voll_kumulativ']) * 100 / population_germany,
    2)
```

```
In [111... vaccinations['fully vaccinated'] = round(
    vaccinations['personen_voll_kumulativ'] * 100 / population_germany,
    2)
```

```
In [112... vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 132 entries, 0 to 131
```

```
Data columns (total 15 columns):
```

#	Column	Non-Null Count	Dtype
0	date	132 non-null	datetime64[ns]
1	dosen_kumulativ	132 non-null	int64
2	dosen_differenz_zum_vortag	132 non-null	int64
3	dosen_erst_differenz_zum_vortag	132 non-null	int64
4	dosen_zweit_differenz_zum_vortag	132 non-null	int64
5	dosen_biontech_kumulativ	132 non-null	int64
6	dosen_moderna_kumulativ	132 non-null	int64
7	dosen_astrazeneca_kumulativ	132 non-null	int64
8	personen_erst_kumulativ	132 non-null	int64
9	personen_voll_kumulativ	132 non-null	int64
10	dosen_dim_kumulativ	132 non-null	int64
11	dosen_kbv_kumulativ	132 non-null	int64
12	dosen_johnson_kumulativ	132 non-null	int64
13	partly vaccinated	132 non-null	float64

14 fully vaccinated 132 non-null float64
 dtypes: datetime64[ns](1), float64(2), int64(12)
 memory usage: 15.6 KB

In [113... vaccinations.tail(3)

Out[113...

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
129	2021-05-05	32652324	1115886	898480	217406	2427062
130	2021-05-06	33594803	942479	737564	204915	2493674
131	2021-05-07	34408840	814037	617514	196523	2548032

Last Update

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

In [114...

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

Out[114... '2021-05-07'

Doses Used

In [115...

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

In [116...

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

Doses Daily

In [117...

```
doses_daily = doses.set_index('date', inplace=False)
doses_daily.tail(1)
```

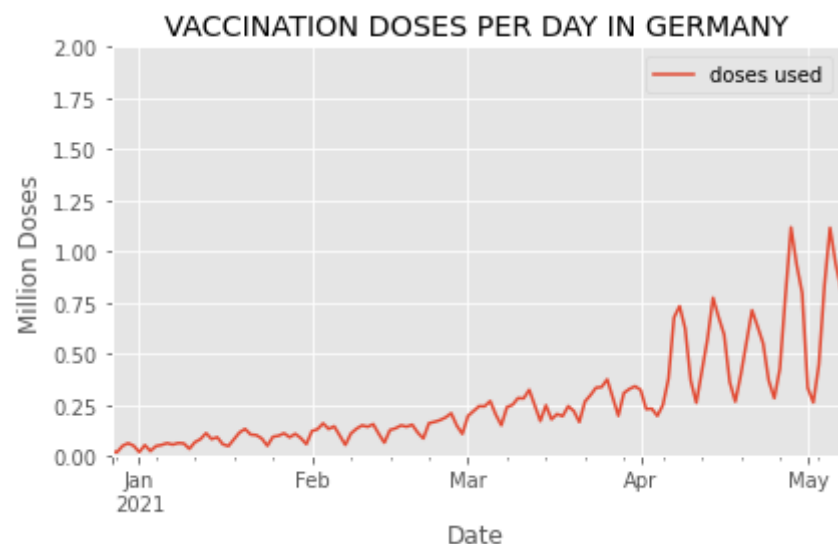
```
Out[117...      doses used  
      date  
2021-05-07    0.814037
```

```
In [118... # What is the highest number of doses used in a day?  
max_doses_daily = max(doses_daily['doses used'])  
max_doses_daily
```

```
Out[118... 1.117913
```

```
In [119... doses_daily.plot(  
    ylim=(0,math.ceil(max_doses_daily)),  
    xlabel='Date',  
    ylabel='Million Doses',  
    title='VACCINATION DOSES PER DAY IN GERMANY')
```

```
Out[119... <AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>
```



Doses per Weekday (in the last 6 weeks)

```
In [120... last_6_weeks = doses.tail(42)
```

```
In [121... # 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-121-45013977109e>: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 [122... # check:
last_6_weeks.tail(3)
```

```
Out[122...      date  doses used  weekday
129  2021-05-05    1.115886  Wednesday
130  2021-05-06    0.942479   Thursday
131  2021-05-07    0.814037    Friday
```

```
In [123... # drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

```
In [124... #last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

```
Out[124...      doses used  weekday
129    1.115886  Wednesday
130    0.942479   Thursday
131    0.814037    Friday
```

```
In [125... pivot_table = last_6_weeks.pivot(columns='weekday', values='doses used')
pivot_table.tail()
```

```
Out[125... weekday  Friday  Monday  Saturday  Sunday  Thursday  Tuesday  Wednesday
127         NaN    0.445775         NaN         NaN         NaN         NaN         NaN
128         NaN         NaN         NaN         NaN         NaN    0.833265         NaN
```

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
129	NaN	NaN	NaN	NaN	NaN	NaN	1.115886
130	NaN	NaN	NaN	NaN	0.942479	NaN	NaN
131	0.814037	NaN	NaN	NaN	NaN	NaN	NaN

In [126...

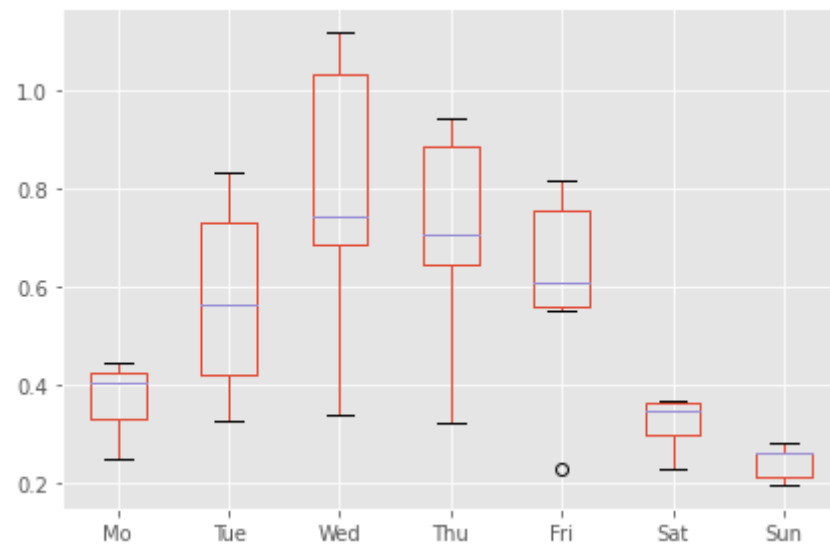
```
# Reorder the columns
pivot_table = pivot_table[['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']]
# Rename the columns
pivot_table.columns=['Mo', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
pivot_table.tail()
```

Out[126...

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
127	0.445775	NaN	NaN	NaN	NaN	NaN	NaN
128	NaN	0.833265	NaN	NaN	NaN	NaN	NaN
129	NaN	NaN	1.115886	NaN	NaN	NaN	NaN
130	NaN	NaN	NaN	0.942479	NaN	NaN	NaN
131	NaN	NaN	NaN	NaN	0.814037	NaN	NaN

In [127...

```
weekday_boxplot = pivot_table.boxplot()
```

```
In [128... fig = weekday_boxplot.get_figure()
fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

```
In [129... # 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[129... million doses used
```

date	
2021-04-12	3.455424
2021-04-19	3.642918
2021-04-26	3.528553
2021-05-03	4.675651
2021-05-10	3.705667

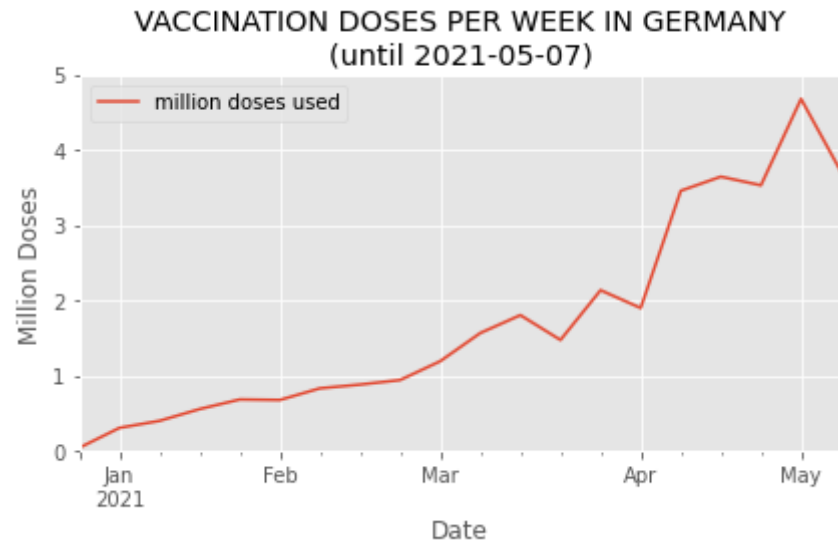
```
In [130... # What is the highest number of doses used in a week?
```

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

Out[130...] 4.675651

```
In [131...] 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[131...] <AxesSubplot:title={'center': 'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-05-07)'}, xlabel='Date', ylabel='Million Doses'>



Doses per Month

```
In [132...] # M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date', freq='M')).sum()
doses_monthly.tail()
```

Out[132...] **doses used**

date	
2021-01-31	2.343091

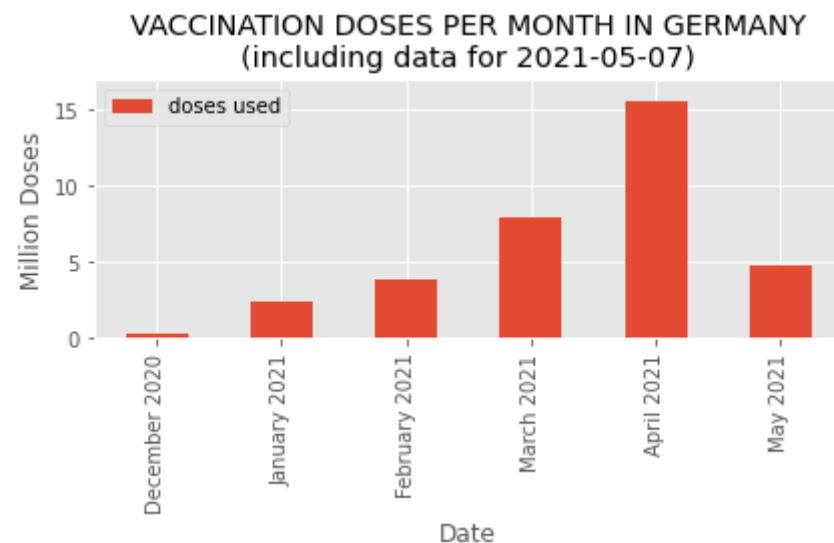
doses used	
date	
2021-02-28	3.776292
2021-03-31	7.845951
2021-04-30	15.492020
2021-05-31	4.746485

```
In [133... 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[133... doses used
```

label	
December 2020	0.205001
January 2021	2.343091
February 2021	3.776292
March 2021	7.845951
April 2021	15.492020
May 2021	4.746485

```
In [134... 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 [135... fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_germany.png')
```

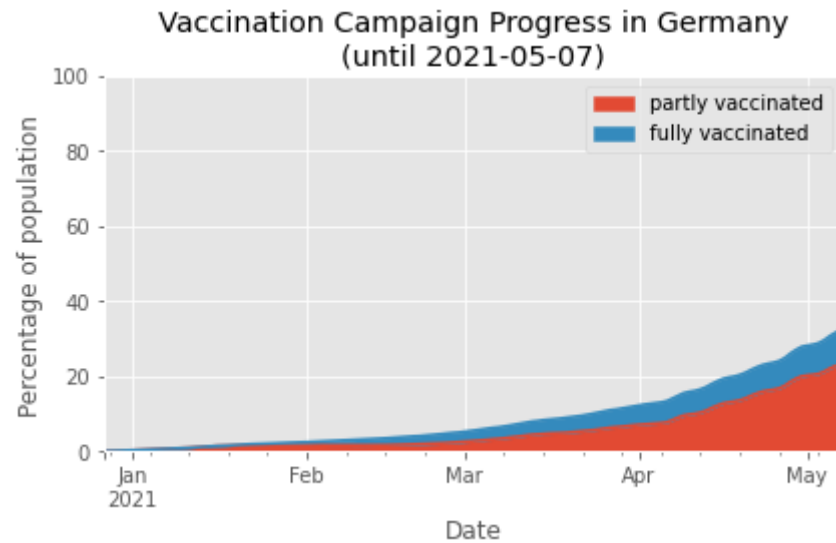
Vaccination Campaign Progress

```
In [136... doses_cumulative = vaccinations.loc[ : , ['date', 'partly vaccinated', 'fully vaccinated']]
doses_cumulative.set_index('date', inplace=True)
doses_cumulative.tail(3)
```

```
Out[136... partly vaccinated  fully vaccinated
```

date		
2021-05-05	22.03	8.62
2021-05-06	22.67	8.87
2021-05-07	23.18	9.10

```
In [137... 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 [138... fig = doses_area_plot.get_figure()
fig.savefig('img/vaccinations_germany_area_plot.png')
```

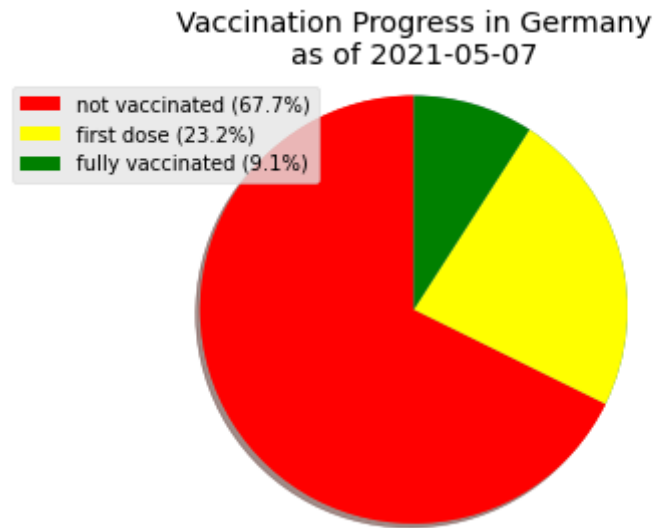
As of Today

```
In [139... # get the last line of the data
current_state = doses_cumulative.iloc[-1]
current_state
```

```
Out[139... partly vaccinated    23.18
fully vaccinated         9.10
Name: 2021-05-07 00:00:00, dtype: float64
```

```
In [140... percentage_not_vacc = 100 - current_state['partly vaccinated'] - current_state['fully vaccinated']
labels = [f"not vaccinated ({round(percentage_not_vacc, 1)}%)",
          f"first dose ({round(current_state['partly vaccinated'], 1)}%)",
          f"fully vaccinated ({round(current_state['fully vaccinated'], 1)}%)"]
colors = ['red', 'yellow', 'green']
sizes = [percentage_not_vacc,
          current_state['partly vaccinated'],
          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 [141... vaccine_use = vaccinations.loc[ : , ['date', 'dosen_biontech_kumulativ',
                                     'dosen_moderna_kumulativ',
                                     'dosen_astrazeneca_kumulativ',
                                     'dosen_johnson_kumulativ']]

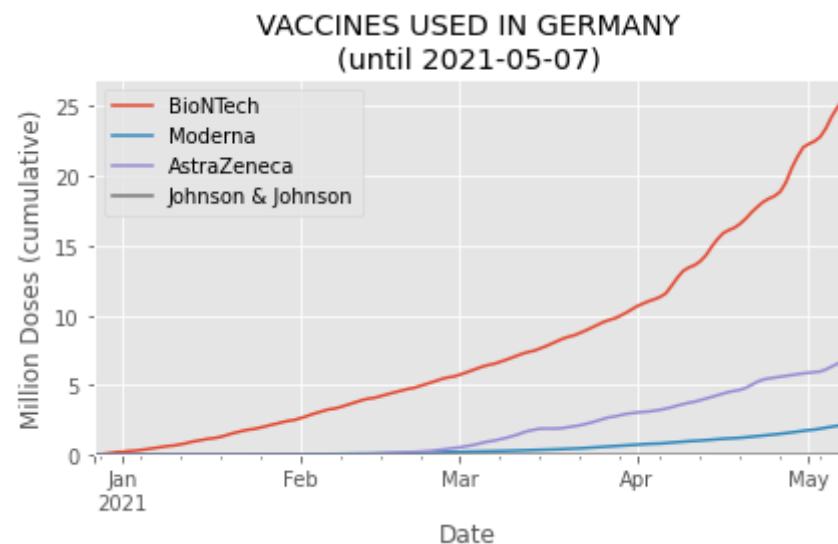
# Rename columns
vaccine_use.columns = ['date', 'BioNTech', 'Moderna', 'AstraZeneca', 'Johnson & Johnson']
# make 'date' an index
vaccine_use.set_index('date', inplace=True)
# divide columns by 1 million
vaccine_use["BioNTech"] = vaccine_use["BioNTech"] / 1_000_000
vaccine_use["Moderna"] = vaccine_use["Moderna"] / 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[141]...

	BioNTech	Moderna	AstraZeneca	Johnson & Johnson
date				
2021-05-05	24.270629	2.021713	6.345802	0.014180
2021-05-06	24.936747	2.098797	6.543104	0.016155
2021-05-07	25.480329	2.164352	6.746226	0.017933

In [142]...

```
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 [143]...

```
fig = vaccines_used.get_figure()
fig.savefig('img/vaccines_used_in_germany.png')
```

Vaccination Centers versus Doctor's Practices

In [144]...

```
by_place = vaccinations.loc[ : , ['date', 'dosen_dim_kumulativ', 'dosen_kbv_kumulativ']]
```

```
by_place.columns = ['date', 'vaccination centers', 'practices']
```

```
In [145... by_place['vaccination centers daily'] = by_place['vaccination centers'].diff()
by_place['practices daily'] = by_place['practices'].diff()
```

```
In [146... by_place['percentage practices'] = round(
    by_place['practices daily'] * 100 /
    (by_place['vaccination centers daily'] + by_place['practices daily']), 2)

by_place['percentage centers'] = 100 - by_place['percentage practices']
```

```
In [147... # make 'date' an index
by_place.set_index('date', inplace=True)
```

```
In [148... by_place
```

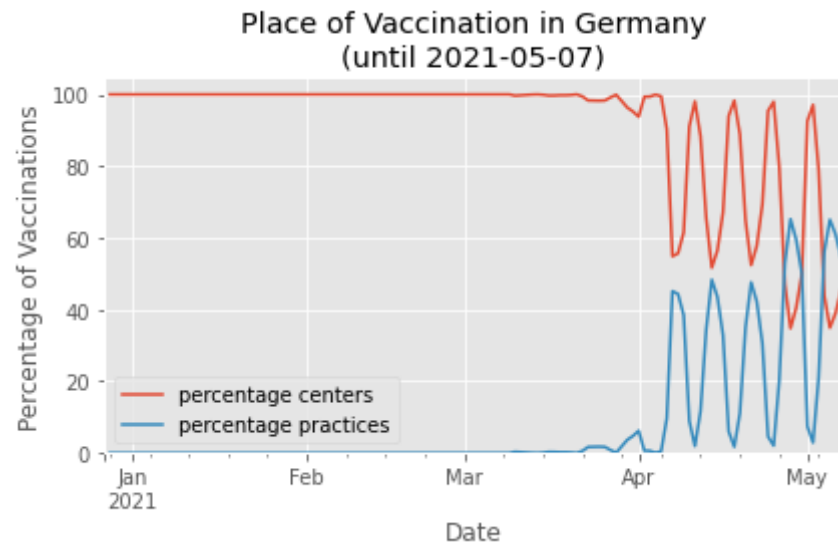
```
Out[148...      vaccination centers  practices  vaccination centers daily  practices daily  percentage practices  percentage centers
date
2020-12-27             24474         0                      NaN                NaN                NaN                NaN
2020-12-28             42816         0             18342.0                0.0              0.00             100.00
2020-12-29             92365         0             49549.0                0.0              0.00             100.00
2020-12-30            154905         0             62540.0                0.0              0.00             100.00
2020-12-31            205001         0             50096.0                0.0              0.00             100.00
...                   ...         ...                   ...                   ...              ...              ...
2021-05-03            25181634      5521539             351915.0             93860.0             21.06             78.94
2021-05-04            25547647      5988791             366013.0             467252.0             56.07             43.93
2021-05-05            25937569      6714755             389922.0             725964.0             65.06             34.94
2021-05-06            26304937      7289866             367368.0             575111.0             61.02             38.98
2021-05-07            26674958      7733882             370021.0             444016.0             54.54             45.46
```

132 rows × 6 columns

```
In [149... share = by_place.loc[:, ['percentage centers', 'percentage practices']]
```



```
In [150... vacc_shares = share.plot(  
    # as it is cumulative, the last row must contain the single highest number  
    ylim=(0, 105), # above 100 to see the line  
    xlabel='Date',  
    ylabel='Percentage of Vaccinations',  
    title=f"Place of Vaccination in Germany\n(until {last_update})")
```



```
In [151... fig = vacc_shares.get_figure()  
fig.savefig('img/vaccinations_germany_by_place.png')
```