

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  
import seaborn
```

Date this Notebook was run

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

```
Out[3]: '2021-05-17'
```

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})
```

```
In [6]: population_germany = 83_200_000
```

Get and Transform Data

```
In [7]: 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 [8]: 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 [9]: 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 [10]: vaccinations.drop(columns=['impf_quote_erst', 'impf_quote_voll'], inplace=True)
```

Convert datatype of date column

```
In [11]: vaccinations.iloc[:, [0]] = vaccinations.iloc[:, [0]].apply(pd.to_datetime)
```

Show Data

```
In [12]: vaccinations.info()

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

RangeIndex: 141 entries, 0 to 140

Data columns (total 13 columns):

#	Column	Non-Null Count	Dtype
0	date	141 non-null	datetime64[ns]
1	dosen_kumulativ	141 non-null	int64
2	dosen_differenz_zum_vortag	141 non-null	int64
3	dosen_erst_differenz_zum_vortag	141 non-null	int64
4	dosen_zweit_differenz_zum_vortag	141 non-null	int64
5	dosen_biontech_kumulativ	141 non-null	int64
6	dosen_moderna_kumulativ	141 non-null	int64
7	dosen_astrazeneca_kumulativ	141 non-null	int64
8	personen_erst_kumulativ	141 non-null	int64
9	personen_voll_kumulativ	141 non-null	int64
10	dosen_dim_kumulativ	141 non-null	int64
11	dosen_kbv_kumulativ	141 non-null	int64
12	dosen_johnson_kumulativ	141 non-null	int64

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

memory usage: 14.4 KB

In [13]: `vaccinations.tail(3)`

Out[13]:

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulativ
138	2021-05-14	39429507	733363	515820	217543	2911806
139	2021-05-15	39806642	377135	225943	151192	2937999
140	2021-05-16	40075392	268750	155602	113148	2957030

Check Validity

In [14]: `# get the last row / the newest available data`
`last_row = vaccinations.tail(1)`

In [15]: `doses_used = last_row['dosen_kumulativ']`
`doses_used`

Out[15]: 140 40075392
 Name: dosen_kumulativ, dtype: int64

```
In [16]: # 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 [17]: # Must be exactly 0
doses_used - partially_vaccinated_people - (fully_vaccinated_people - johnson_doses) * 2 - johnson_doses == 0
```

```
Out[17]: 140    True
dtype: bool
```

Calculate columns

```
In [18]: vaccinations['partly vaccinated'] = round(
    (vaccinations['personen_erst_kumulativ'] - vaccinations['personen_voll_kumulativ']) * 100 / population_germany,
    2)
```

```
In [19]: vaccinations['fully vaccinated'] = round(
    vaccinations['personen_voll_kumulativ'] * 100 / population_germany,
    2)
```

```
In [20]: vaccinations.info()
```

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

```
RangeIndex: 141 entries, 0 to 140
```

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

#	Column	Non-Null Count	Dtype
0	date	141 non-null	datetime64[ns]
1	dosen_kumulativ	141 non-null	int64
2	dosen_differenz_zum_vortag	141 non-null	int64
3	dosen_erst_differenz_zum_vortag	141 non-null	int64
4	dosen_zweit_differenz_zum_vortag	141 non-null	int64
5	dosen_biontech_kumulativ	141 non-null	int64
6	dosen_moderna_kumulativ	141 non-null	int64
7	dosen_astrazeneca_kumulativ	141 non-null	int64
8	personen_erst_kumulativ	141 non-null	int64
9	personen_voll_kumulativ	141 non-null	int64
10	dosen_dim_kumulativ	141 non-null	int64
11	dosen_kbv_kumulativ	141 non-null	int64
12	dosen_johnson_kumulativ	141 non-null	int64
13	partly vaccinated	141 non-null	float64

```

14 fully vaccinated          141 non-null    float64
dtypes: datetime64[ns](1), float64(2), int64(12)
memory usage: 16.6 KB

```

```
In [21]: vaccinations.tail(3)
```

```
Out[21]:
```

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
138	2021-05-14	39429507	733363	515820	217543	2911806
139	2021-05-15	39806642	377135	225943	151192	2937999
140	2021-05-16	40075392	268750	155602	113148	2957030

Last Update

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

```
In [22]: last_update = vaccinations.loc[vaccinations.index[-1], "date"].strftime('%Y-%m-%d')
last_update
```

```
Out[22]: '2021-05-16'
```

Doses Used

```
In [23]: doses = vaccinations.loc[:, ['date', 'dosen_differenz_zum_vortag']]
# Rename columns
doses.columns = ['date', 'doses used']
```

```
In [24]: # Scale number of doses as millions
doses['doses used'] = doses['doses used'] / 1_000_000
```

Doses Daily

```
In [25]: doses_daily = doses.set_index('date', inplace=False)
doses_daily.tail(1)
```

Out[25]:

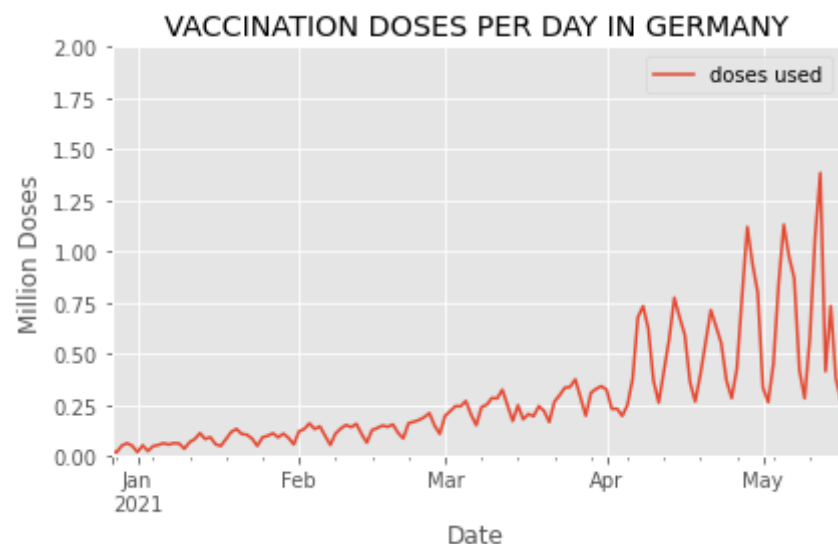
	doses used
	date
	2021-05-16
	0.26875

```
In [26]: # What is the highest number of doses used in a day?
max_doses_daily = max(doses_daily['doses used'])
max_doses_daily
```

Out[26]: 1.384331

```
In [27]: doses_daily.plot(
    ylim=(0,math.ceil(max_doses_daily)),
    xlabel='Date',
    ylabel='Million Doses',
    title='VACCINATION DOSES PER DAY IN GERMANY')
```

Out[27]: <AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>



Doses per Weekday (in the last 6 weeks)

```
In [28]: last_6_weeks = doses.tail(42)
```

```
In [29]: # 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-29-45013977109e>: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 [30]: # check:
last_6_weeks.tail(3)
```

```
Out[30]:
```

	date	doses used	weekday
138	2021-05-14	0.733363	Friday
139	2021-05-15	0.377135	Saturday
140	2021-05-16	0.268750	Sunday

```
In [31]: # drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

```
In [32]: #last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

```
Out[32]:
```

	doses used	weekday
138	0.733363	Friday
139	0.377135	Saturday
140	0.268750	Sunday

```
In [33]: pivot_table = last_6_weeks.pivot(columns='weekday', values='doses used')
pivot_table.tail()
```

```
Out[33]:
```

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
136	NaN	NaN	NaN	NaN	NaN	NaN	1.384331
137	NaN	NaN	NaN	NaN	0.413854	NaN	NaN

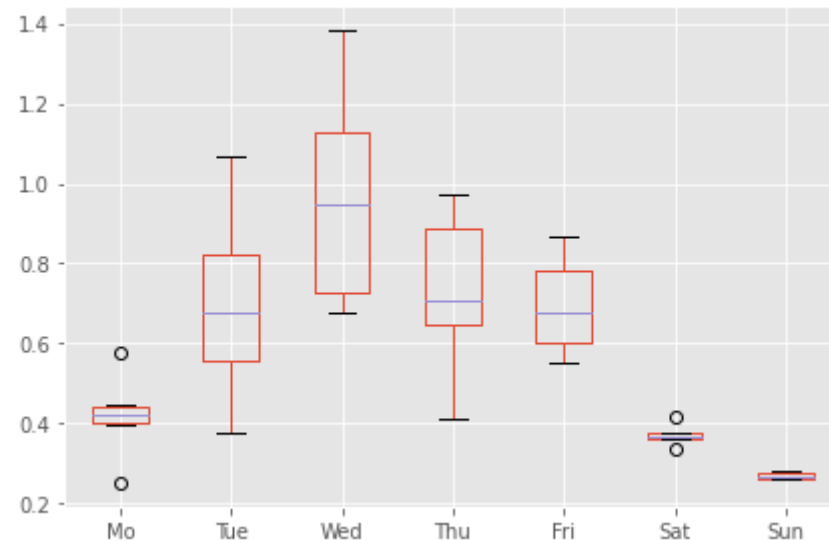
weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
138	0.733363	NaN	NaN	NaN	NaN	NaN	NaN
139	NaN	NaN	0.377135	NaN	NaN	NaN	NaN
140	NaN	NaN	NaN	0.26875	NaN	NaN	NaN

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

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
136	NaN	NaN	1.384331	NaN	NaN	NaN	NaN
137	NaN	NaN	NaN	0.413854	NaN	NaN	NaN
138	NaN	NaN	NaN	NaN	0.733363	NaN	NaN
139	NaN	NaN	NaN	NaN	NaN	0.377135	NaN
140	NaN	NaN	NaN	NaN	NaN	NaN	0.26875

```
In [35]: weekday_boxplot = pivot_table.boxplot()
```

```
In [36]: fig = weekday_boxplot.get_figure()
fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

```
In [37]: # 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[37]:
```

million doses used	
date	
2021-04-19	3.644751
2021-04-26	3.533388
2021-05-03	4.689089
2021-05-10	5.093775
2021-05-17	4.246627

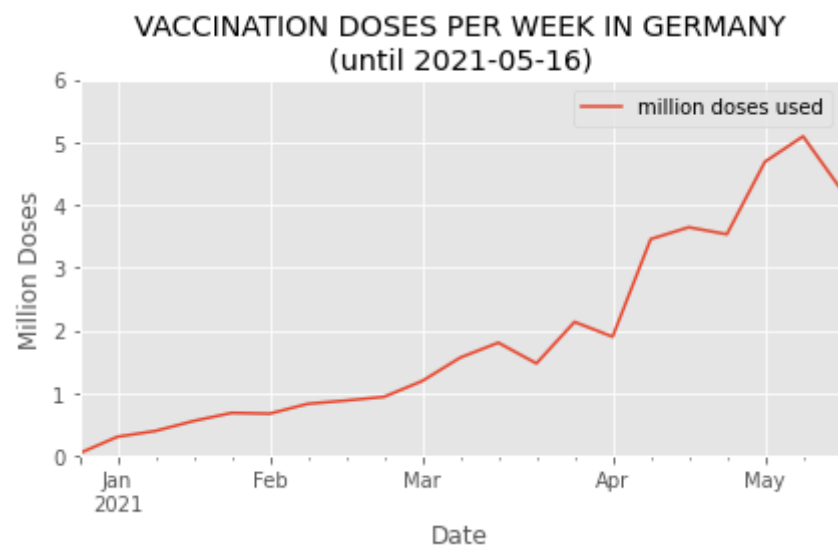
```
In [38]: # What is the highest number of doses used in a week?
```

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

Out[38]: 5.093775

```
In [39]: 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[39]: <AxesSubplot:title={'center': 'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-05-16)'}, xlabel='Date', ylabel='Million Doses'>



Doses per Month

```
In [40]: # M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date', freq='M')).sum()
doses_monthly.tail()
```

Out[40]:

doses used	
date	
2021-01-31	2.343264

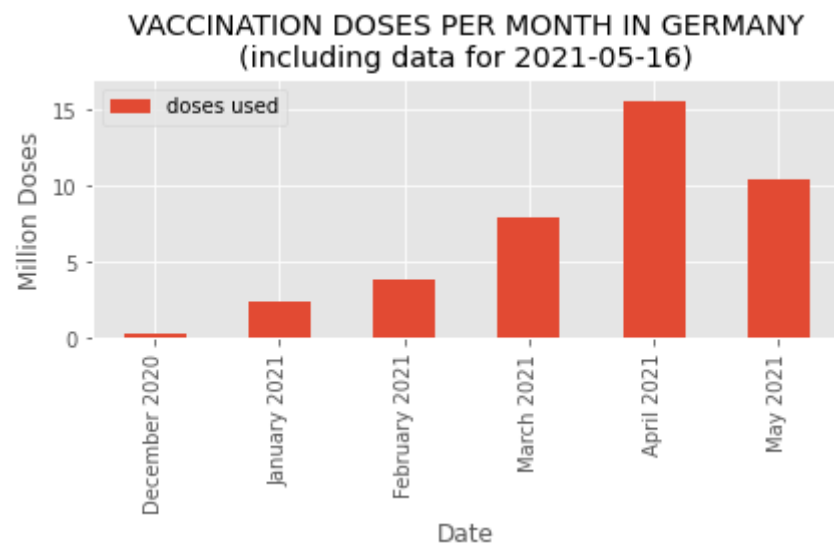
doses used	
date	
2021-02-28	3.778666
2021-03-31	7.851925
2021-04-30	15.508643
2021-05-31	10.386919

```
In [41]: 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[41]:
```

doses used	
label	
December 2020	0.205975
January 2021	2.343264
February 2021	3.778666
March 2021	7.851925
April 2021	15.508643
May 2021	10.386919

```
In [42]: 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 [43]: fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_germany.png')
```

Vaccination Campaign Progress

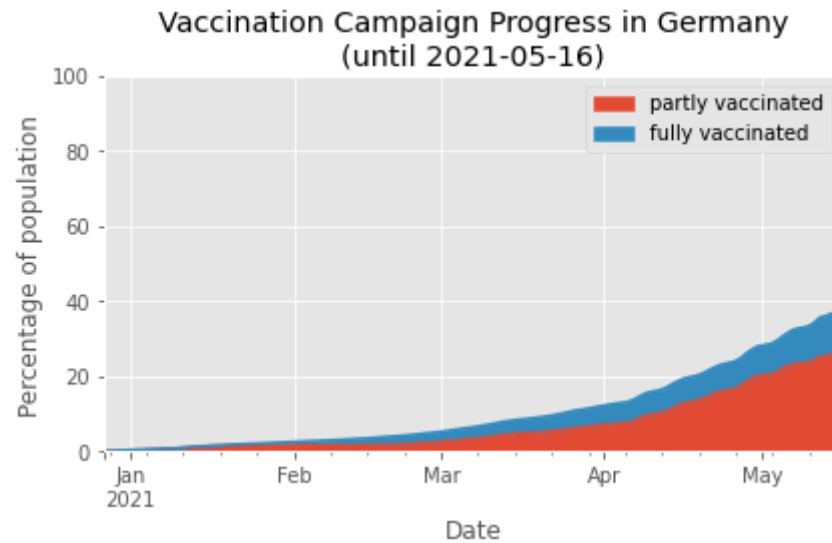
```
In [44]: doses_cumulative = vaccinations.loc[:, ['date', 'partly vaccinated', 'fully vaccinated']]
doses_cumulative.set_index('date', inplace=True)
doses_cumulative.tail(3)
```

```
Out[44]:
```

	partly vaccinated	fully vaccinated
date		

date		
2021-05-14	25.64	10.90
2021-05-15	25.74	11.08
2021-05-16	25.79	11.22

```
In [45]: 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 [46]: fig = doses_area_plot.get_figure()
fig.savefig('img/vaccinations_germany_area_plot.png')
```

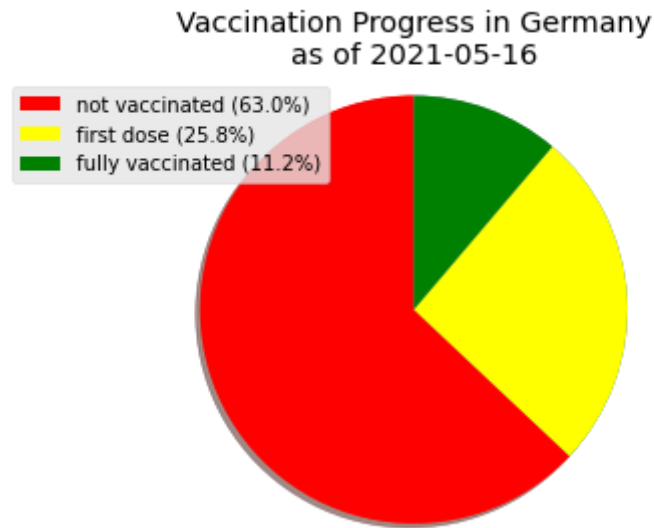
As of Today

```
In [47]: # get the last line of the data
current_state = doses_cumulative.iloc[-1]
current_state
```

```
Out[47]: partly vaccinated    25.79
fully vaccinated             11.22
Name: 2021-05-16 00:00:00, dtype: float64
```

```
In [48]: 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 [49]: vaccine_use = vaccinations.loc[ : , ['date', 'dosen_biontech_kumulativ',
                                              'dosen_moderna_kumulativ',
                                              'dosen_astrazeneca_kumulativ',
                                              'dosen_johnson_kumulativ']]

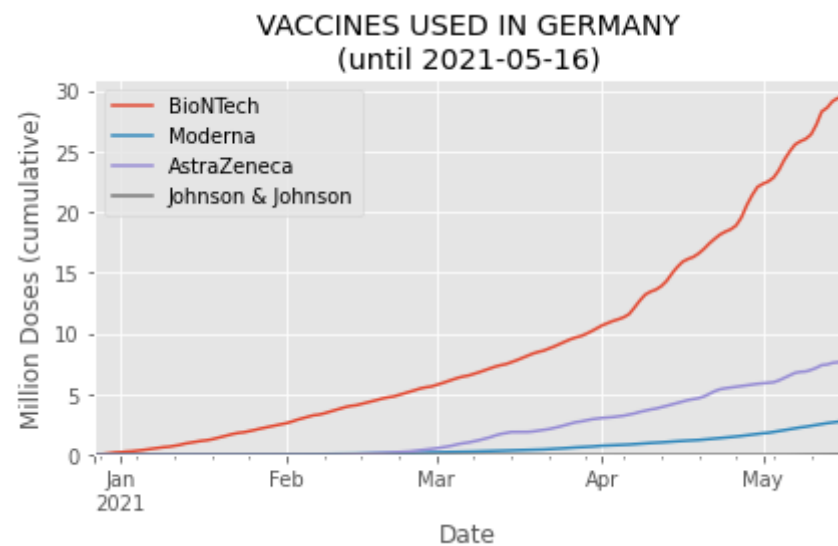
# Rename columns
vaccine_use.columns = ['date', 'BioNTech', 'Moderna', 'AstraZeneca', 'Johnson & Johnson']
# make 'date' an index
vaccine_use.set_index('date', inplace=True)
# divide columns by 1 million
vaccine_use["BioNTech"] = vaccine_use["BioNTech"] / 1_000_000
vaccine_use["Moderna"] = vaccine_use["Moderna"] / 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[49]:

	BioNTech	Moderna	AstraZeneca	Johnson & Johnson
date				
2021-05-14	29.118067	2.685209	7.586342	0.039889
2021-05-15	29.379992	2.742167	7.641089	0.043394
2021-05-16	29.570308	2.792232	7.665114	0.047738

In [50]:

```
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 [51]:

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

Vaccination Centers versus Doctor's Practices

In [52]:

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

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

```
In [53]: by_place['vaccination centers daily'] = by_place['vaccination centers'].diff()
by_place['practices daily'] = by_place['practices'].diff()
```

```
In [54]: 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 [55]: # make 'date' an index
by_place.set_index('date', inplace=True)
```

```
In [56]: by_place
```

```
Out[56]:
```

	vaccination centers	practices	vaccination centers daily	practices daily	percentage practices	percentage centers
--	---------------------	-----------	---------------------------	-----------------	----------------------	--------------------

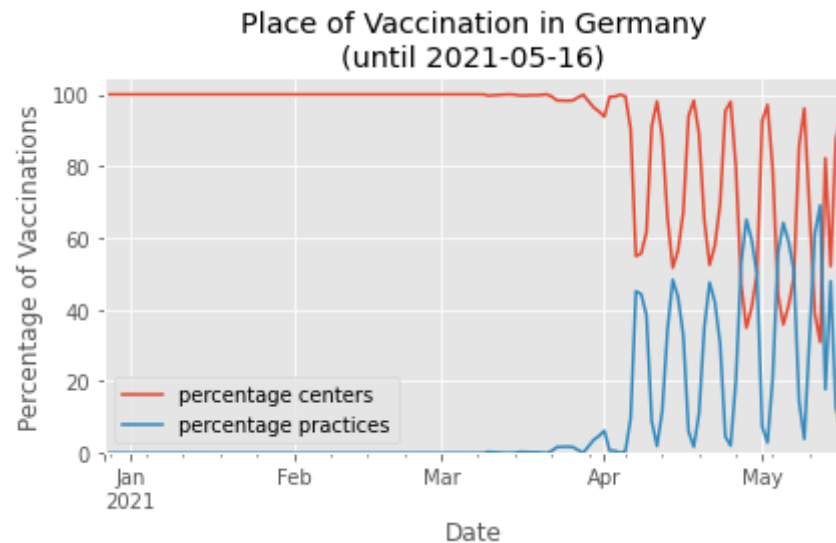
date						
2020-12-27	24000	0	NaN	NaN	NaN	NaN
2020-12-28	42559	0	18559.0	0.0	0.00	100.00
2020-12-29	93289	0	50730.0	0.0	0.00	100.00
2020-12-30	155956	0	62667.0	0.0	0.00	100.00
2020-12-31	205975	0	50019.0	0.0	0.00	100.00
...
2021-05-12	28678306	9603984	428348.0	955983.0	69.06	30.94
2021-05-13	29018681	9677463	340375.0	73479.0	17.75	82.25
2021-05-14	29400707	10028800	382026.0	351337.0	47.91	52.09
2021-05-15	29730176	10076466	329469.0	47666.0	12.64	87.36
2021-05-16	29982921	10092471	252745.0	16005.0	5.96	94.04

141 rows × 6 columns

```
In [57]: share = by_place.loc[:, ['percentage centers', 'percentage practices']]
```



```
In [58]: 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 [59]: fig = vacc_shares.get_figure()
fig.savefig('img/vaccinations_germany_by_place.png')
```

Other units of Time

```
In [60]: by_place_daily = by_place.loc[ : , ['vaccination centers daily', 'practices daily']]
by_place_daily.columns = ['vaccination centers', 'practices']
by_place_daily.reset_index(inplace=True)
```

Monthly

```
In [61]: by_place_monthly = by_place_daily.groupby(pd.Grouper(key='date', freq='M')).sum()
by_place_monthly.tail()
```

```
Out[61]:
```

	vaccination centers	practices
date		

	vaccination centers	practices
date		
2021-01-31	2343264.0	0.0
2021-02-28	3778666.0	0.0
2021-03-31	7785691.0	66234.0
2021-04-30	10179503.0	5329140.0
2021-05-31	5689822.0	4697097.0

Scale:

```
In [62]: by_place_monthly['vaccination centers'] = by_place_monthly['vaccination centers'] / 1_000_000
by_place_monthly['practices'] = by_place_monthly['practices'] / 1_000_000
```

Rename the columns

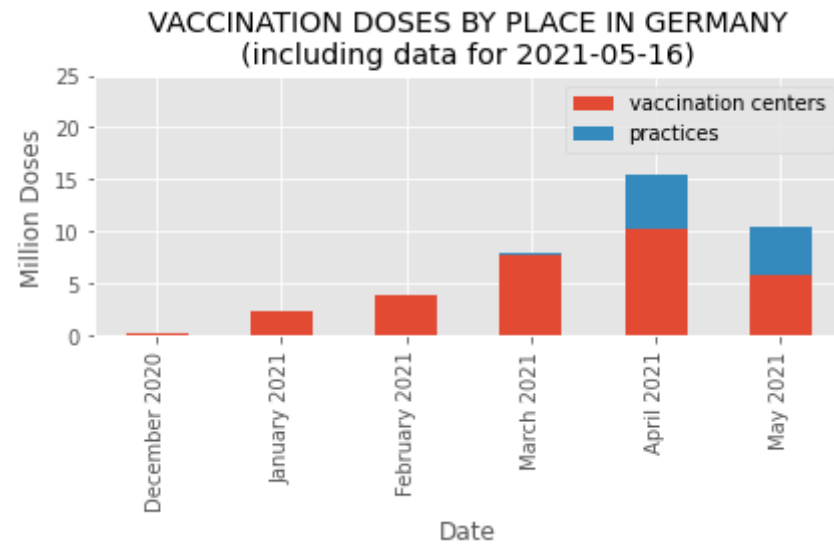
```
In [63]: by_place_monthly['month'] = by_place_monthly.index.strftime('%B')
by_place_monthly['year'] = by_place_monthly.index.strftime('%Y')
by_place_monthly['label'] = by_place_monthly['month'] + ' ' + by_place_monthly['year']
by_place_monthly.drop(columns=['month', 'year'], inplace=True)
by_place_monthly.set_index('label', inplace=True)
by_place_monthly.tail(6)
```

```
Out[63]:
```

	vaccination centers	practices
label		
December 2020	0.181975	0.000000
January 2021	2.343264	0.000000
February 2021	3.778666	0.000000
March 2021	7.785691	0.066234
April 2021	10.179503	5.329140
May 2021	5.689822	4.697097

```
In [64]: monthly_plot = by_place_monthly.plot.bar(
stacked=True,
```

```
ylim=(0, 25),  
xlabel='Date',  
ylabel='Million Doses',  
title=f"VACCINATION DOSES BY PLACE IN GERMANY\n(including data for {last_update})")
```



```
In [65]: fig = monthly_plot.get_figure()  
fig.savefig('img/monthly_doses_by_place_germany.png')
```