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 [100…
```
# standard library
import datetime
import math
```

In [101…
```
# 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 [102…
```
today = datetime.datetime.today().strftime('%Y-%m-%d')
today
```

Out[102… '2021-05-12'

Set Defaults

In [103…
```
# style like ggplot in R
plt.style.use('ggplot')
```

In [104…
```
# 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 [105…
```
population_germany = 83_200_000
```

Get and Transform Data

In [106…
```
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 [107…
```
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 [108…
```
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 [109…
```
vaccinations.drop(columns=['impf_quote_erst', 'impf_quote_voll'], inplace=True)
```

Convert datatype of date column

In [110…
```
vaccinations.iloc[ : , [0]] = vaccinations.iloc[ : , [0]].apply(pd.to_datetime)
```

Show Data

In [111…
```
vaccinations.info()
```

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

```
RangeIndex: 136 entries, 0 to 135
Data columns (total 13 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   date                            136 non-null    datetime64[ns]
 1   dosen_kumulativ                 136 non-null    int64
 2   dosen_differenz_zum_vortag      136 non-null    int64
 3   dosen_erst_differenz_zum_vortag 136 non-null    int64
 4   dosen_zweit_differenz_zum_vortag 136 non-null   int64
 5   dosen_biontech_kumulativ        136 non-null    int64
 6   dosen_moderna_kumulativ         136 non-null    int64
 7   dosen_astrazeneca_kumulativ     136 non-null    int64
 8   personen_erst_kumulativ         136 non-null    int64
 9   personen_voll_kumulativ         136 non-null    int64
 10  dosen_dim_kumulativ             136 non-null    int64
 11  dosen_kbv_kumulativ             136 non-null    int64
 12  dosen_johnson_kumulativ         136 non-null    int64
dtypes: datetime64[ns](1), int64(12)
memory usage: 13.9 KB
```

In [112…

```
vaccinations.tail(3)
```

Out[112…

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
133	2021-05-09	35222512	276612	177042	99570	2600759
134	2021-05-10	35789319	566807	384810	181997	2636693
135	2021-05-11	36837184	1047865	782301	265564	2716866

Check Validity

In [113…

```
# get the last row / the newest available data
last_row = vaccinations.tail(1)
```

In [114…

```
doses_used = last_row['dosen_kumulativ']
doses_used
```

Out[114…

```
135    36837184
Name: dosen_kumulativ, dtype: int64
```

In [115…
```
# 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 [116…
```
# Must be exactly 0
doses_used - partially_vaccinated_people - (fully_vaccinated_people - johnson_doses) * 2 - johnson_doses ==  0
```

Out[116…
```
135    True
dtype: bool
```

Calculate columns

In [117…
```
vaccinations['partly vaccinated'] = round(
    (vaccinations['personen_erst_kumulativ'] - vaccinations['personen_voll_kumulativ']) * 100 / population_germany,
    2)
```

In [118…
```
vaccinations['fully vaccinated'] = round(
    vaccinations['personen_voll_kumulativ'] * 100 / population_germany,
    2)
```

In [119…
```
vaccinations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 136 entries, 0 to 135
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype         
---  ------                          --------------  -----         
 0   date                            136 non-null    datetime64[ns]
 1   dosen_kumulativ                 136 non-null    int64         
 2   dosen_differenz_zum_vortag      136 non-null    int64         
 3   dosen_erst_differenz_zum_vortag 136 non-null    int64         
 4   dosen_zweit_differenz_zum_vortag 136 non-null    int64         
 5   dosen_biontech_kumulativ        136 non-null    int64         
 6   dosen_moderna_kumulativ         136 non-null    int64         
 7   dosen_astrazeneca_kumulativ     136 non-null    int64         
 8   personen_erst_kumulativ         136 non-null    int64         
 9   personen_voll_kumulativ         136 non-null    int64         
 10  dosen_dim_kumulativ             136 non-null    int64         
 11  dosen_kbv_kumulativ             136 non-null    int64         
 12  dosen_johnson_kumulativ         136 non-null    int64         
 13  partly vaccinated               136 non-null    float64
```

```
 14  fully vaccinated                  136 non-null    float64
dtypes: datetime64[ns](1), float64(2), int64(12)
memory usage: 16.1 KB
```

In [120…
```
vaccinations.tail(3)
```

Out[120…

	date	dosen_kumulativ	dosen_differenz_zum_vortag	dosen_erst_differenz_zum_vortag	dosen_zweit_differenz_zum_vortag	dosen_biontech_kumulati
133	2021-05-09	35222512	276612	177042	99570	2600759
134	2021-05-10	35789319	566807	384810	181997	2636693
135	2021-05-11	36837184	1047865	782301	265564	2716866

Last Update

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

In [121…
```
last_update = vaccinations.loc[vaccinations.index[-1], "date"].strftime('%Y-%m-%d')
last_update
```

Out[121… `'2021-05-11'`

Doses Used

In [122…
```
doses = vaccinations.loc[ : , ['date', 'dosen_differenz_zum_vortag']]
# Rename columns
doses.columns = ['date', 'doses used']
```

In [123…
```
# Scale number of doses as millions
doses['doses used'] = doses['doses used'] / 1_000_000
```

Doses Daily

In [124…
```
doses_daily = doses.set_index('date', inplace=False)
doses_daily.tail(1)
```

Out[124…

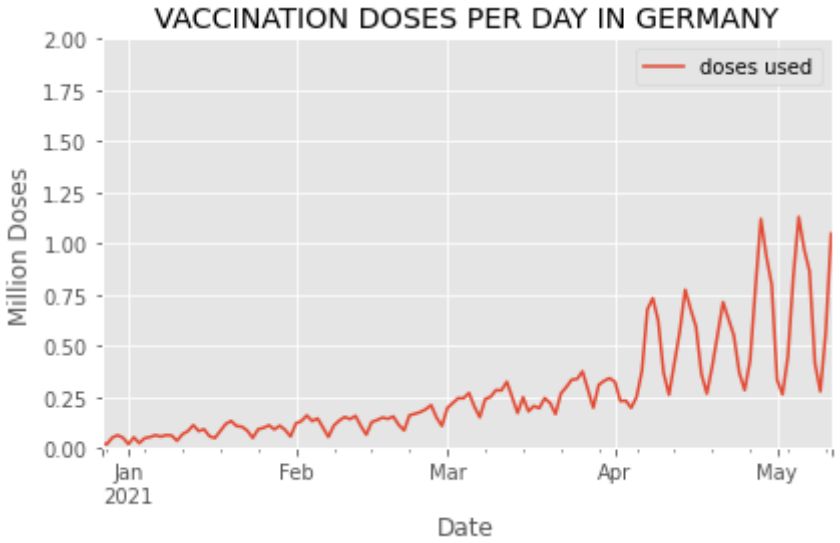
date	doses used
2021-05-11	1.047865

In [125…
```
# What is the highest number of doses used in a day?
max_doses_daily = max(doses_daily['doses used'])
max_doses_daily
```

Out[125… 1.129985

In [126…
```
doses_daily.plot(
    ylim=(0,math.ceil(max_doses_daily)),
    xlabel='Date',
    ylabel='Million Doses',
    title='VACCINATION DOSES PER DAY IN GERMANY')
```

Out[126… <AxesSubplot:title={'center':'VACCINATION DOSES PER DAY IN GERMANY'}, xlabel='Date', ylabel='Million Doses'>



Doses per Weekday (in the last 6 weeks)

In [127…
```
last_6_weeks = doses.tail(42)
```

In [128…
```
# 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-128-45013977109e>: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 [129…
```
# check:
last_6_weeks.tail(3)
```

Out[129…

	date	doses used	weekday
133	2021-05-09	0.276612	Sunday
134	2021-05-10	0.566807	Monday
135	2021-05-11	1.047865	Tuesday

In [130…
```
# drop the date column
last_6_weeks = last_6_weeks.drop(labels=['date'], axis=1)
```

In [131…
```
#last_6_weeks.set_index('weekday', inplace=True)
last_6_weeks.tail(3)
```

Out[131…

	doses used	weekday
133	0.276612	Sunday
134	0.566807	Monday
135	1.047865	Tuesday

In [132…
```
pivot_table =last_6_weeks.pivot(columns='weekday', values='doses used')
pivot_table.tail()
```

Out[132…

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
131	0.86439	NaN	NaN	NaN	NaN	NaN	NaN
132	NaN	NaN	0.415372	NaN	NaN	NaN	NaN

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
133	NaN	NaN	NaN	0.276612	NaN	NaN	NaN
134	NaN	0.566807	NaN	NaN	NaN	NaN	NaN
135	NaN	NaN	NaN	NaN	NaN	1.047865	NaN

In [133…

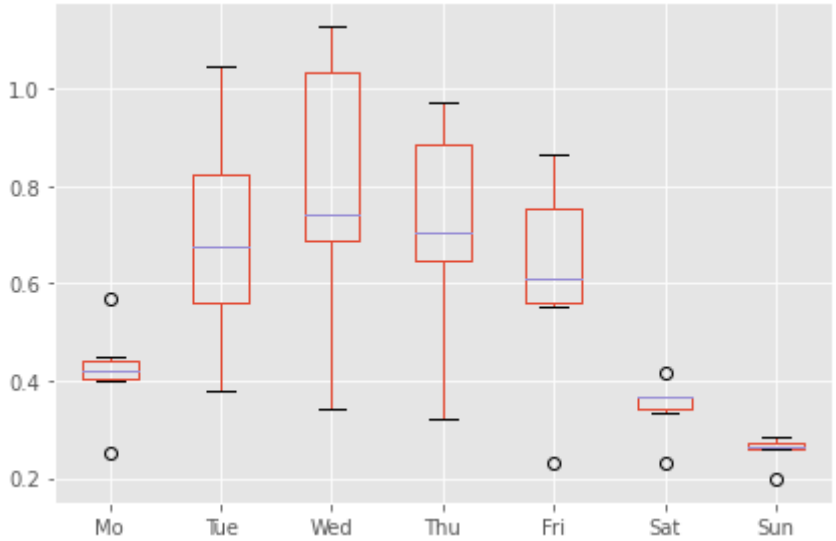
```
# Reorder the columns
pivot_table = pivot_table[['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']]
# Rename the columns
pivot_table.columns=['Mo', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
pivot_table.tail()
```

Out[133…

	Mo	Tue	Wed	Thu	Fri	Sat	Sun
131	NaN	NaN	NaN	NaN	0.86439	NaN	NaN
132	NaN	NaN	NaN	NaN	NaN	0.415372	NaN
133	NaN	NaN	NaN	NaN	NaN	NaN	0.276612
134	0.566807	NaN	NaN	NaN	NaN	NaN	NaN
135	NaN	1.047865	NaN	NaN	NaN	NaN	NaN

In [134…

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



In [135…
```
fig = weekday_boxplot.get_figure()
fig.savefig('img/weekday_boxplot.png')
```

Doses per Week

In [136…
```
# 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[136…

date	million doses used
2021-04-19	3.644413
2021-04-26	3.531381
2021-05-03	4.685294
2021-05-10	5.063456
2021-05-17	1.047865

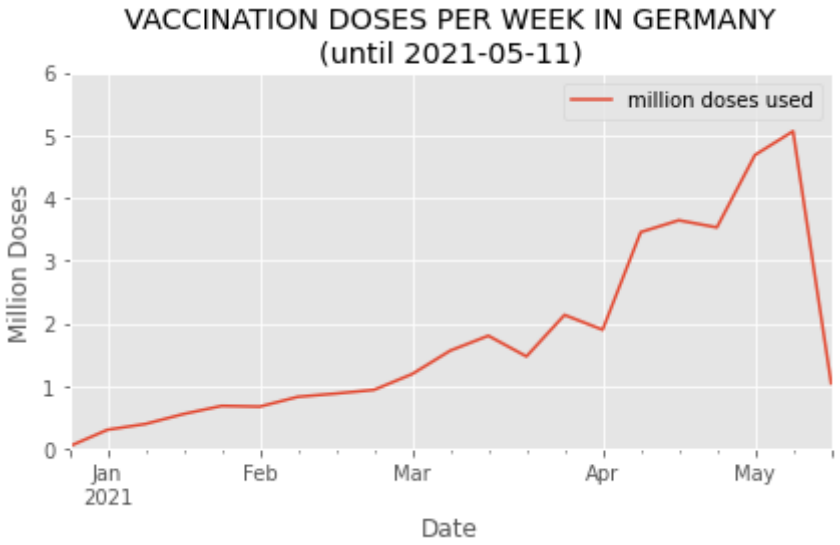
In [137…
```
# What is the highest number of doses used in a week?
```

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

Out[137… 5.0634559999999995

In [138…
```
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[138… <AxesSubplot:title={'center':'VACCINATION DOSES PER WEEK IN GERMANY\n(until 2021-05-11)'}, xlabel='Date', ylabel='Million Doses'>



Doses per Month

In [139…
```
# M = month end frequency
doses_monthly = doses.groupby(pd.Grouper(key='date',freq='M')).sum()
doses_monthly.tail()
```

Out[139…

	doses used
date	
2021-01-31	2.343200

date	doses used
2021-02-28	3.778409
2021-03-31	7.850496
2021-04-30	15.504202
2021-05-31	7.154967

In [140…

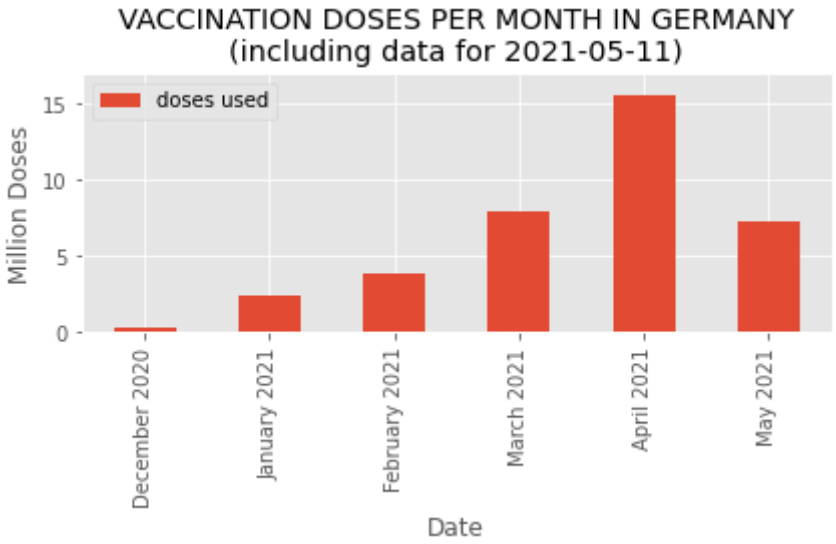
```
max_doses_monthly = max(doses_monthly['doses used'])
max_doses_monthly
doses_monthly['month'] = doses_monthly.index.strftime('%B')
doses_monthly['year'] =  doses_monthly.index.strftime('%Y')
doses_monthly['label'] = doses_monthly['month'] + ' ' + doses_monthly['year']
doses_monthly.drop(columns=['month', 'year'], inplace=True)
doses_monthly.set_index('label', inplace=True)
doses_monthly.tail(6)
```

Out[140…

label	doses used
December 2020	0.205910
January 2021	2.343200
February 2021	3.778409
March 2021	7.850496
April 2021	15.504202
May 2021	7.154967

In [141…

```
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 [142…
```
fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_germany.png')
```

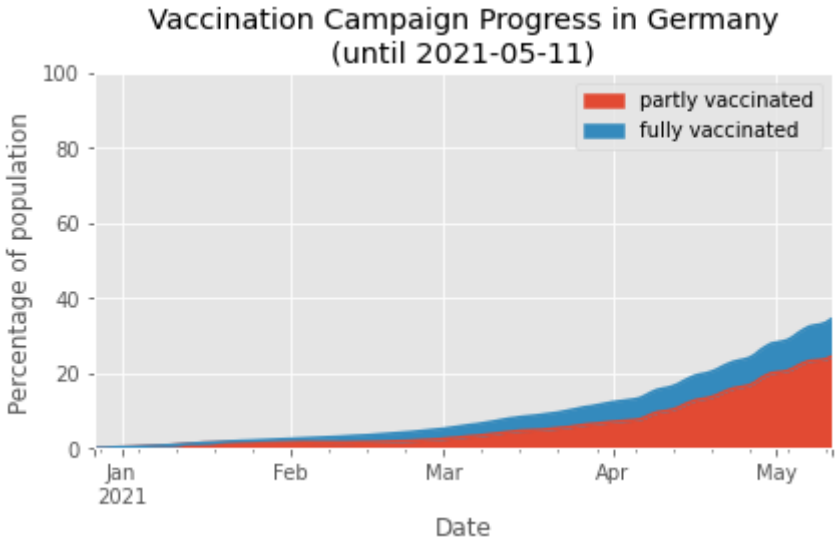
Vaccination Campaign Progress

In [143…
```
doses_cumulative = vaccinations.loc[ : , ['date', 'partly vaccinated', 'fully vaccinated']]
doses_cumulative.set_index('date', inplace=True)
doses_cumulative.tail(3)
```

Out[143…

date	partly vaccinated	fully vaccinated
2021-05-09	23.44	9.46
2021-05-10	23.68	9.68
2021-05-11	24.31	10.00

In [144…
```
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 [145…
```
fig = doses_area_plot.get_figure()
fig.savefig('img/vaccinations_germany_area_plot.png')
```

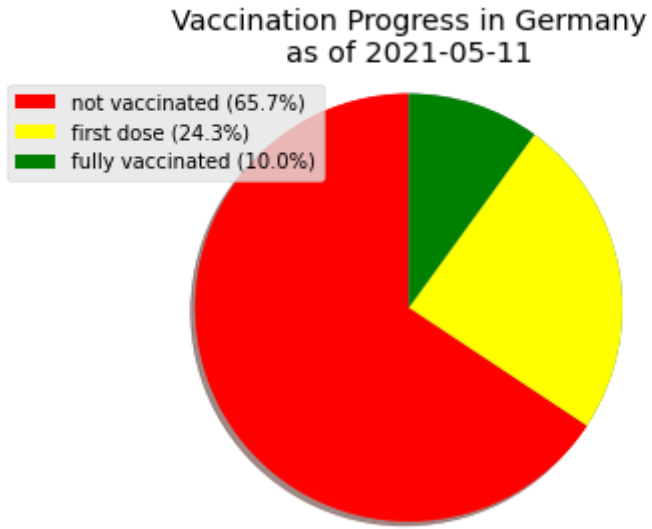
As of Today

In [146…
```
# get the last line of the data
current_state = doses_cumulative.iloc[-1]
current_state
```

Out[146…
```
partly vaccinated    24.31
fully vaccinated     10.00
Name: 2021-05-11 00:00:00, dtype: float64
```

In [147…
```
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 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 [148…

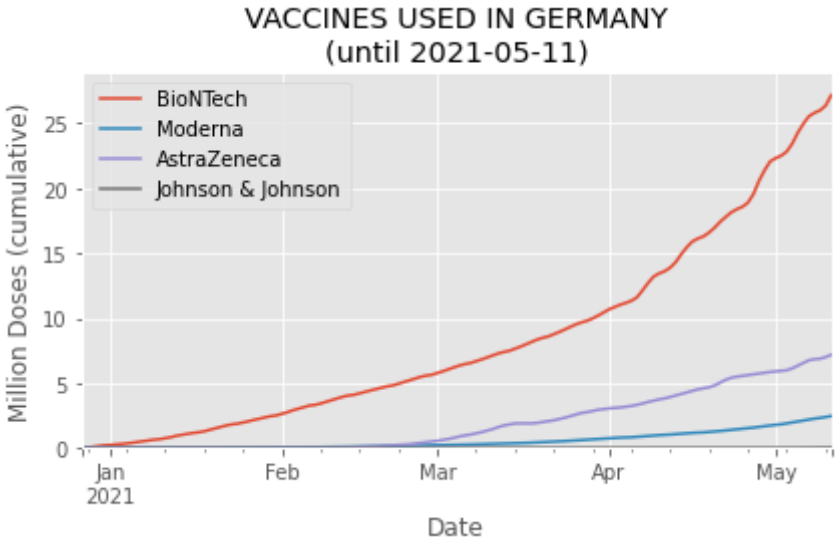
```
vaccine_use = vaccinations.loc[ : , ['date', 'dosen_biontech_kumulativ',
                                     'dosen_moderna_kumulativ',
                                     'dosen_astrazeneca_kumulativ',
                                     'dosen_johnson_kumulativ']]
# Rename columns
vaccine_use.columns = ['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[148…

date	BioNTech	Moderna	AstraZeneca	Johnson & Johnson
2021-05-09	26.007599	2.317416	6.875393	0.022104
2021-05-10	26.366932	2.393300	7.003765	0.025322
2021-05-11	27.168667	2.463861	7.176720	0.027936

In [149…

```
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 [150…

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

Vaccination Centers versus Doctor’s Practices

In [151…

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

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

In [152…
```
by_place['vaccination centers daily'] = by_place['vaccination centers'].diff()
by_place['practices daily'] = by_place['practices'].diff()
```

In [153…
```
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 [154…
```
# make 'date' an index
by_place.set_index('date', inplace=True)
```

In [155…
```
by_place
```

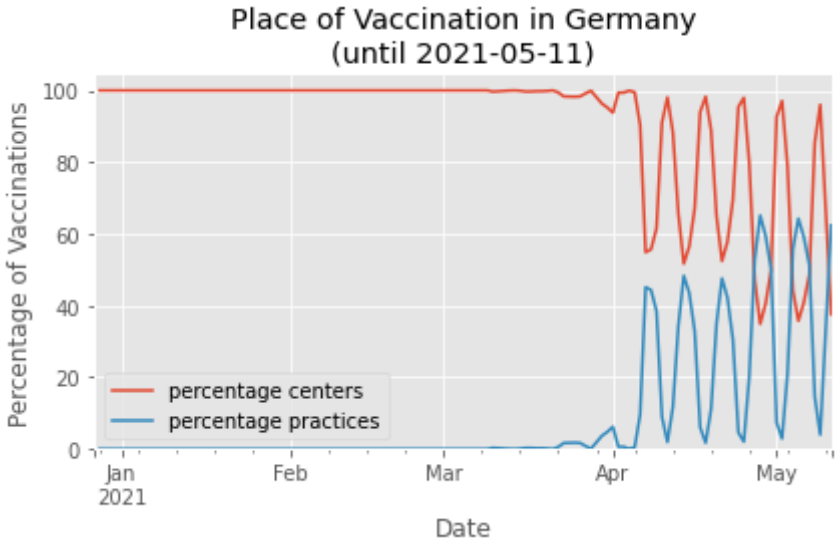
Out[155…

date	vaccination centers	practices	vaccination centers daily	practices daily	percentage practices	percentage centers
2020-12-27	23999	0	NaN	NaN	NaN	NaN
2020-12-28	42484	0	18485.0	0.0	0.00	100.00
2020-12-29	93219	0	50735.0	0.0	0.00	100.00
2020-12-30	155891	0	62672.0	0.0	0.00	100.00
2020-12-31	205910	0	50019.0	0.0	0.00	100.00
...	...	...	...	...	...	...
2021-05-07	26796646	7733882	420374.0	444016.0	51.37	48.63
2021-05-08	27151527	7794373	354881.0	60491.0	14.56	85.44
2021-05-09	27417104	7805408	265577.0	11035.0	3.99	96.01
2021-05-10	27795013	7994306	377909.0	188898.0	33.33	66.67
2021-05-11	28189183	8648001	394170.0	653695.0	62.38	37.62

136 rows × 6 columns

In [156…
```
share = by_place.loc[ : , ['percentage centers', 'percentage practices']]
```

In [157…
```
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 [158…
```
fig = vacc_shares.get_figure()
fig.savefig('img/vaccinations_germany_by_place.png')
```

Other units of Time

In [159…
```
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 [160…
```
by_place_monthly = by_place_daily.groupby(pd.Grouper(key='date',freq='M')).sum()
by_place_monthly.tail()
```

Out[160…

date	vaccination centers	practices

date	vaccination centers	practices
2021-01-31	2343200.0	0.0
2021-02-28	3778409.0	0.0
2021-03-31	7784262.0	66234.0
2021-04-30	10175062.0	5329140.0
2021-05-31	3902340.0	3252627.0

Scale:

In [161…
```
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

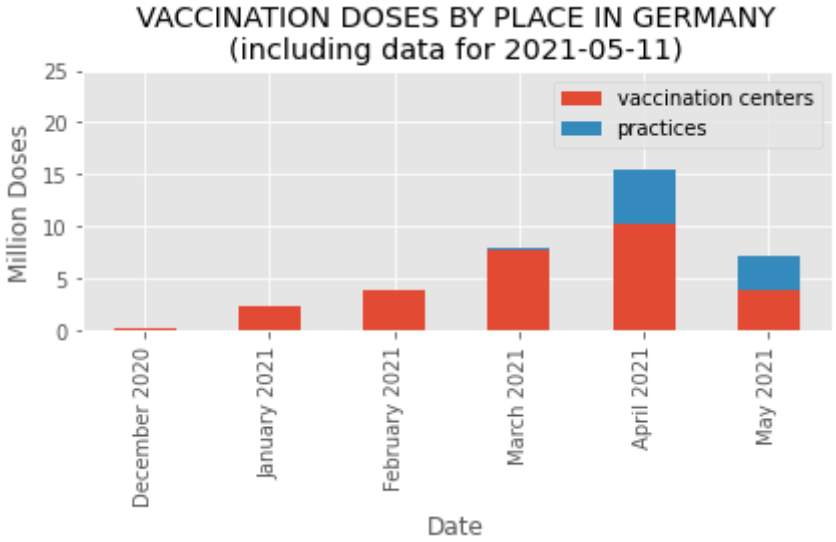
In [162…
```
by_place_monthly['month'] = by_place_monthly.index.strftime('%B')
by_place_monthly['year'] =  by_place_monthly.index.strftime('%Y')
by_place_monthly['label'] = by_place_monthly['month'] + ' ' + by_place_monthly['year']
by_place_monthly.drop(columns=['month', 'year'], inplace=True)
by_place_monthly.set_index('label', inplace=True)
by_place_monthly.tail(6)
```

Out[162…

label	vaccination centers	practices
December 2020	0.181911	0.000000
January 2021	2.343200	0.000000
February 2021	3.778409	0.000000
March 2021	7.784262	0.066234
April 2021	10.175062	5.329140
May 2021	3.902340	3.252627

In [163…
```
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 [164…

```
fig = monthly_plot.get_figure()
fig.savefig('img/monthly_doses_by_place_germany.png')
```