Step 1 - Data from CDC

November 15, 2021

Table of Contents

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

1 Read data from CDC

[1]:
```
# Key data is hardcoded.

import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import requests
from sodapy import Socrata
client = Socrata('data.cdc.gov',
                 '',
                 username='',
                 password='')

#results = client.get("8xkx-amqh", limit = 100000)
results = client.get("unsk-b7fc", limit = 150000)
vaccines = pd.DataFrame(results)

# This is where the documentation is:
# https://data.cdc.gov/Vaccinations/
 ↪COVID-19-Vaccinations-in-the-United-States-Jurisdi/unsk-b7fc
```

[2]:
```
print(vaccines.columns)
print(vaccines.shape)
```

```
Index(['date', 'mmwr_week', 'location', 'distributed', 'distributed_janssen',
       'distributed_moderna', 'distributed_pfizer', 'distributed_unk_manuf',
       'dist_per_100k', 'distributed_per_100k_12plus',
       'distributed_per_100k_18plus', 'distributed_per_100k_65plus',
       'administered', 'administered_12plus', 'administered_18plus',
       'administered_65plus', 'administered_janssen', 'administered_moderna',
       'administered_pfizer', 'administered_unk_manuf', 'admin_per_100k',
       'admin_per_100k_12plus', 'admin_per_100k_18plus',
       'admin_per_100k_65plus', 'recip_administered',
       'administered_dose1_recip', 'administered_dose1_pop_pct',
       'administered_dose1_recip_1', 'administered_dose1_recip_2',
       'administered_dose1_recip_3', 'administered_dose1_recip_4',
       'administered_dose1_recip_5', 'administered_dose1_recip_6',
       'series_complete_yes', 'series_complete_pop_pct',
       'series_complete_12plus', 'series_complete_12pluspop',
       'series_complete_18plus', 'series_complete_18pluspop',
       'series_complete_65plus', 'series_complete_65pluspop',
       'series_complete_janssen', 'series_complete_moderna',
       'series_complete_pfizer', 'series_complete_unk_manuf',
       'series_complete_janssen_12plus', 'series_complete_moderna_12plus',
       'series_complete_pfizer_12plus', 'series_complete_unk_manuf_1',
       'series_complete_janssen_18plus', 'series_complete_moderna_18plus',
       'series_complete_pfizer_18plus', 'series_complete_unk_manuf_2',
       'series_complete_janssen_65plus', 'series_complete_moderna_65plus',
       'series_complete_pfizer_65plus', 'series_complete_unk_manuf_3',
       'additional_doses', 'additional_doses_vax_pct',
       'additional_doses_18plus', 'additional_doses_18plus_vax_pct',
       'additional_doses_50plus', 'additional_doses_50plus_vax_pct',
       'additional_doses_65plus', 'additional_doses_65plus_vax_pct',
       'additional_doses_moderna', 'additional_doses_pfizer',
       'additional_doses_janssen', 'additional_doses_unk_manuf'],
      dtype='object')
(21720, 69)
```

2 Create a 2nd df to manipulate

2.1 Create a subset based on date

[ ]:
```
v2 = vaccines[vaccines.date >= '2021-04-27']
v2.shape

# At this point, date is an object not a date
```

```
[ ]: # Create a new dataframe with just the required columns
     # new=  old[['A', 'C', 'D']]
     v2 =␣
      ↪vaccines[['date','mmwr_week','location','distributed','administered','distributed_janssen',
         'distributed_moderna','distributed_pfizer',␣
      ↪'additional_doses','administered_12plus',
                  ␣
      ↪'administered_18plus','administered_65plus','series_complete_yes','series_complete_12plus',
                   'series_complete_18plus','series_complete_65plus']]

[ ]: v2.head()

[ ]: v2.info()

[ ]: # Drop columns
     drop_columns = {'distributed_janssen',
         'distributed_moderna',
         'distributed_pfizer'}

     v2 = v2.drop(columns = drop_columns)

[ ]: # Which columns have null values?

     v2.isnull().sum()

[ ]: # Fill the Null values with zero

     v2['additional_doses'] = v2['additional_doses'].fillna(0)
     # Alternate code v2['additional_doses'].fillna(0, inplace = True)

     v2.isnull().sum()

[ ]: #. Change the datatypes

     v2['date'] = v2['date'].astype('datetime64[ns]')

     v2['distributed'] = pd.to_numeric(v2['distributed']).astype(int)
     v2['administered'] = pd.to_numeric(v2['administered']).astype(int)
     v2['additional_doses'] = pd.to_numeric(v2['additional_doses']).astype(int)
     v2['administered_12plus'] = pd.to_numeric(v2['administered_12plus']).astype(int)
     v2['administered_18plus'] = pd.to_numeric(v2['administered_18plus']).astype(int)
     v2['administered_65plus'] = pd.to_numeric(v2['administered_65plus']).astype(int)
     v2['series_complete_yes'] = pd.to_numeric(v2['series_complete_yes']).astype(int)
     v2['series_complete_12plus'] = pd.to_numeric(v2['series_complete_12plus']).
      ↪astype(int)
     v2['series_complete_18plus'] = pd.to_numeric(v2['series_complete_18plus']).
      ↪astype(int)
```

```
v2['series_complete_65plus'] = pd.to_numeric(v2['series_complete_65plus']).
↪astype(int)
```

[ ]:
```
v2.info()
```

[ ]:
```
#Create year, month, day columns from date

v2['year']= v2['date'].dt.year
v2['month']= v2['date'].dt.month
v2['day']= v2['date'].dt.day
v2.head()
```

[ ]:
```
# Create a column to classify the distribution volume

v2['volume'] = pd.cut(v2['distributed'],␣
↪bins=[0,100000,1000000,100000000],labels=['low','medium','high'],right =␣
↪False)
v2.head()
```

3 Create a df with just data from Delaware

[ ]:
```
v2DE = v2[v2.location == 'DE']
```

[ ]:
```
v2DE = v2DE.sort_values('date')
```

[ ]:
```
v2DE.shape
```

[ ]:
```
v2DE.head()
```

[ ]:
```
# How many vaccines were distributed at the beginning of the month?
v2DE['dist_first'] = v2DE.sort_values(by=['day']).groupby(['year',␣
↪'month'])['distributed'].transform('first')
v2DE['dist_last']  = v2DE.sort_values(by=['day']).groupby(['year',␣
↪'month'])['distributed'].transform('last')
```

[ ]:
```
v2DE.head()
```

[ ]:
```
v2DE[['date', 'mmwr_week','distributed','year','month','day','dist_first',␣
↪'dist_last']]
```

4 Create a df that contains data summarized by year and month

[ ]:
```
# Only aggregate distributed and administered

v2_agg = v2.groupby(['year','month']).agg({'distributed':
↪['sum','mean'],'administered':['sum','mean']}).reset_index()
```

```
v2_agg.head()

# df_new = df.groupby(['col1', 'col2'])["col3", "col4"].sum()
```

4.0.1 What needs to change if we want to the same for v2DE but we want to exclude the mean and include dist_first, dist_last?

[ ]:
```
v2DE2 = v2DE.groupby(['year','month']).agg(distributed = ('distributed','sum'),
                                           administered =␣
 ↪('administered','sum'),
                                           dist_first = ('dist_first','min'),
                                           dist_last = ('dist_last','min')).
 ↪reset_index()
v2DE2.head()
```

[ ]:
```
#v2_DE2['prev_last'] = v2_DE2.sort_values(by=['month']).
 ↪groupby(['year'])['dist_last','min'].shift(1)
```

5 Write out each df as a csv file

[ ]:
```
v2.to_csv('vaccines_delivered.csv', index = False)
v2DE.to_csv('DE vaccines delivered.csv', index = False)
v2_agg.to_csv('v2 aggregated.csv', index = False)
```