Using GroupBy with CDC data

June 23, 2022

1 Using GroupBy with CDC data

[ ]:
```
import requests
import numpy as py
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
# Get the data from CDC and look at it in json format

response = requests.get("https://data.cdc.gov/resource/saz5-9hgg.json")
jsonhold = response.json()
jsonhold
```

[ ]:
```
# Put the data into a DataFrame

vaccines = pd.DataFrame(jsonhold)
vaccines
```

[ ]:
```
# Create month and week columns

vaccines['month'] = pd.to_datetime(vaccines['week_of_allocations']).dt.month
vaccines['week'] =  pd.to_datetime(vaccines['week_of_allocations']).dt.week
vaccines
```

[ ]:
```
# This is just to show that the qty is not numeric
# Groupby will find all numeric columns (unless otherwise specified) and␣
 ↪aggregate them

vaccines.groupby('jurisdiction').sum()
```

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

[ ]:
```
# Changing the datatypes & column names

vaccines['month'] = vaccines.month.astype(str)
vaccines['week'] = vaccines.week.astype(str)
```

```
vaccines['_1st_dose_allocations'] = pd.
 ↪to_numeric(vaccines['_1st_dose_allocations']).astype(int)
vaccines['_2nd_dose_allocations'] = pd.
 ↪to_numeric(vaccines['_2nd_dose_allocations']).astype(int)
vaccines['_2nd_dose_allocations'] = vaccines._2nd_dose_allocations*1.2


short_names = {'_1st_dose_allocations':'first',
               '_2nd_dose_allocations':'second'}
vaccines.rename(columns=short_names, inplace=True)

vaccines.info()
```

[ ]:
```
vaccines.jurisdiction.unique()
```

[ ]:
```
vaccines = vaccines[vaccines.jurisdiction.isin(['Massachusetts','New␣
 ↪Hampshire', 'Rhode Island'])]
vaccines.shape
```

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

1.1 Using matplotlib

[ ]:
```
# We cannot.  Matplotlib does not work with dataframes.

plt.bar(vaccines.month,vaccines.second)
```

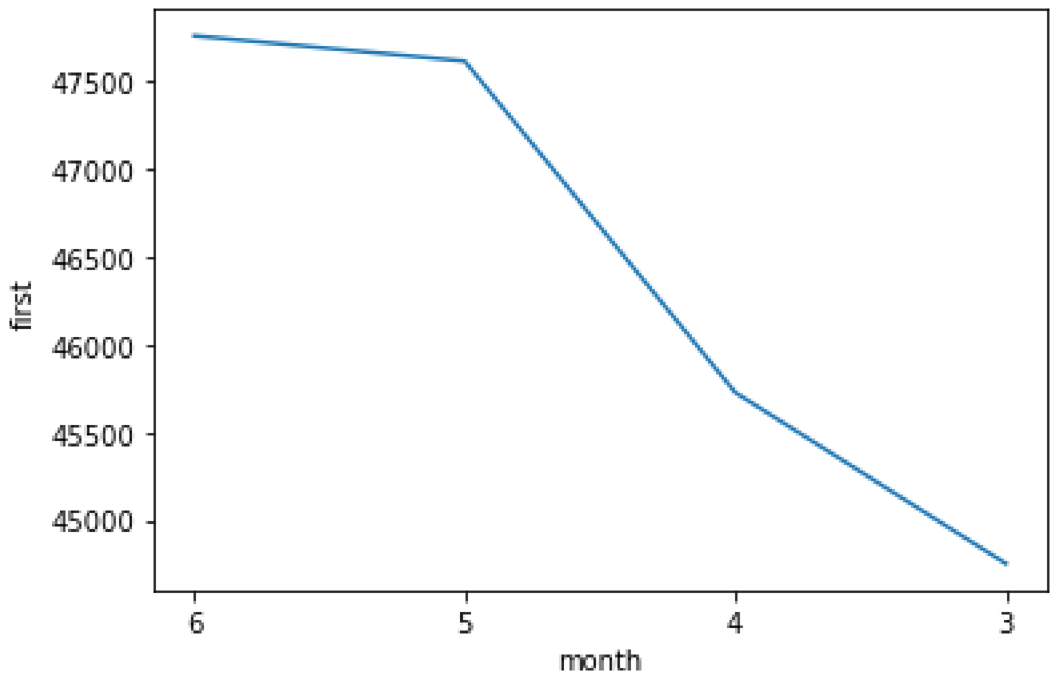
1.2 Using seaborn

[48]:
```
vaccines.groupby('month').mean()
```

[48]:
```
         first   second
month
3      44752.5  53703.0
4      45727.5  54873.0
5      47616.0  57139.2
6      47760.0  57312.0
```

[45]:
```
sns.lineplot(data=vaccines, x='month', y='first', ci = None)
# Seaborn has aggregated the data but uses mean as the aggregation
```

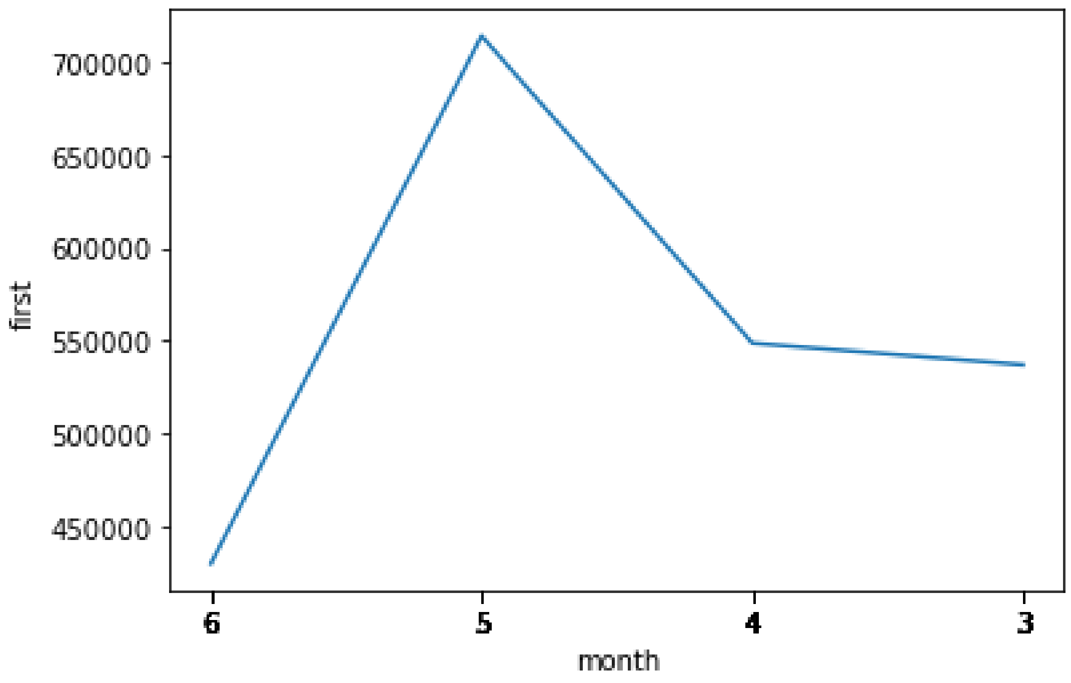
[45]:
```
<AxesSubplot:xlabel='month', ylabel='first'>
```



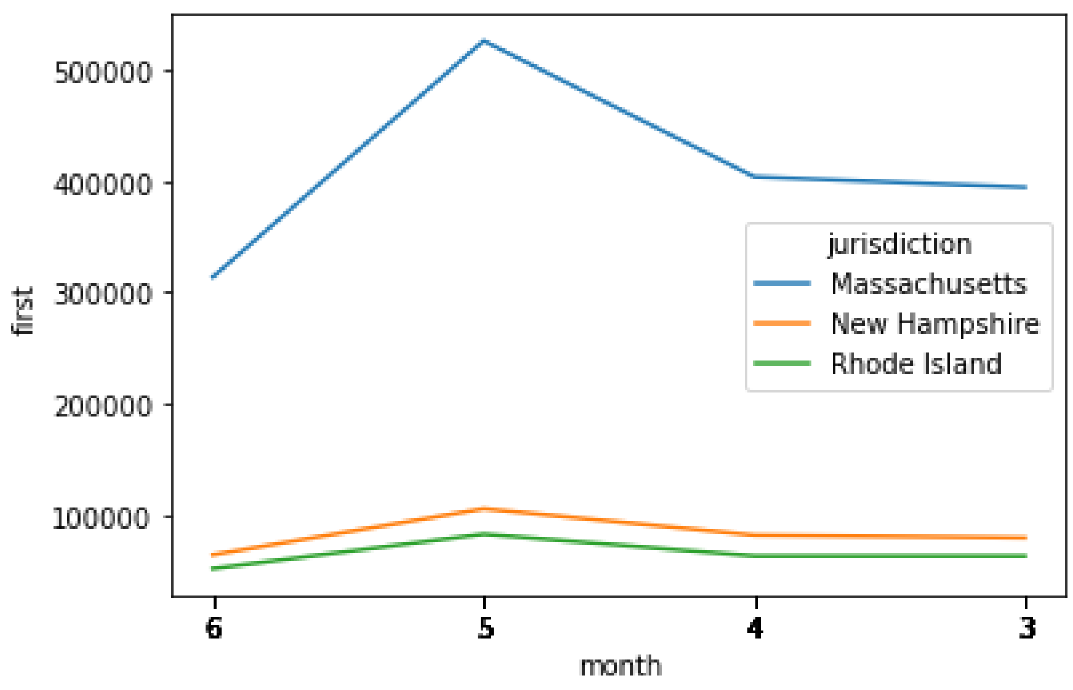
```
[49]: vaccines.groupby('month').sum()
```

```
[49]:         first    second
      month
      3      537030  644436.0
      4      548730  658476.0
      5      714240  857088.0
      6      429840  515808.0
```

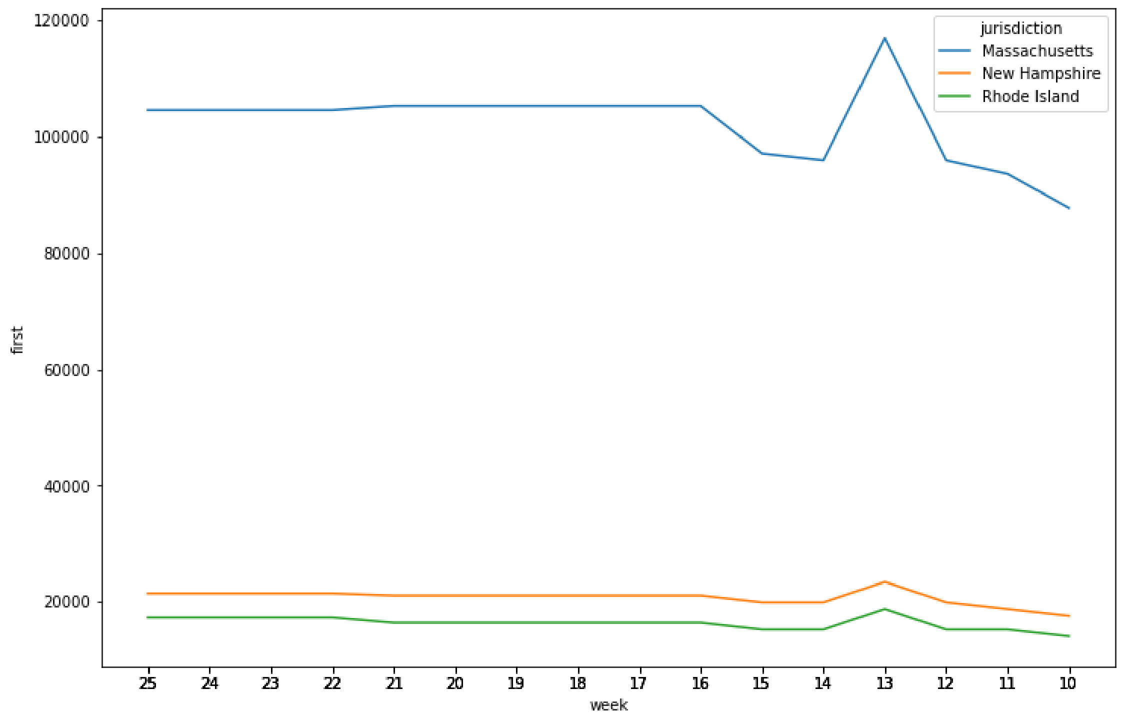
```
[50]: sns.lineplot(data=vaccines, x='month', y='first', ci = None, estimator = 'sum')
      plt.xticks(vaccines.month)
      plt.show()
```



```
[51]: # Add a line for each jurisdiction
sns.lineplot(data=vaccines, x='month', y='first', ci = None, estimator = 'sum',␣
  ↪hue = 'jurisdiction')
plt.xticks(vaccines.month)
plt.show()
```



```
[52]: # Change month to week
      plt.figure(figsize=(12,8))
      sns.lineplot(data=vaccines, x='week', y='first', ci = None, estimator = 'sum',␣
        ↪hue = 'jurisdiction')
      plt.xticks(vaccines.week)
      plt.show()
```

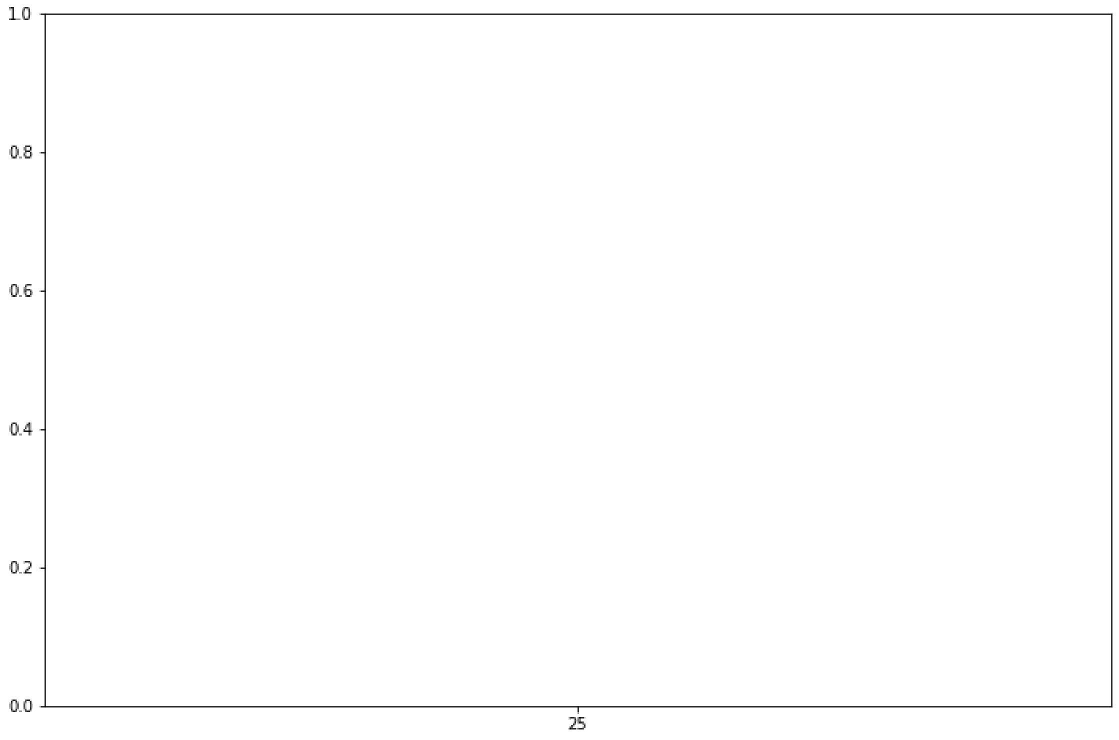


```
[53]: # Add a facetgrid
      # Cannot do that with lineplot
      plt.figure(figsize=(12,8))
      sns.lineplot(data=vaccines, x='week', y='first', ci = None, estimator = 'sum',␣
       ↪hue = 'jurisdiction',col='month')
      plt.xticks(vaccines.week)
      plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [53], in <cell line: 4>()
      1 # Add a facetgrid
      2 # Cannot do that with lineplot
      3 plt.figure(figsize=(12,8))
----> 4␣
 ↪sns.lineplot(data=vaccines, x='week', y='first', ci = None, estimator = 'sum', hue = 'juris
      5 plt.xticks(vaccines.week)
      6 plt.show()

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/seaborn/_decorators.py:
 ↪46, in _deprecate_positional_args.<locals>.inner_f(*args, **kwargs)
     36     warnings.warn(
     37         "Pass the following variable{} as {}keyword arg{}: {}. "
     38         "From version 0.12, the only valid positional argument "
```

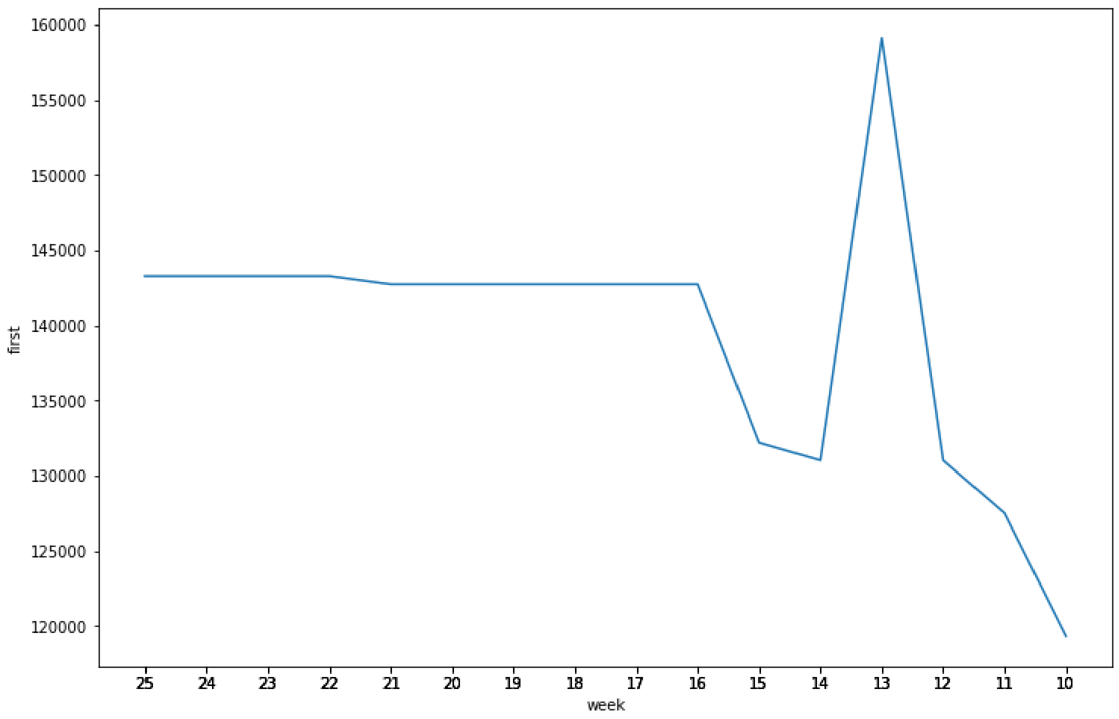
```
   (…)
     43         FutureWarning
     44     )
     45 kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46 return f(**kwargs)

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/seaborn/relational.py
 ↪710, in lineplot(x, y, hue, size, style, data, palette, hue_order, hue_norm,␣
 ↪sizes, size_order, size_norm, dashes, markers, style_order, units, estimator,␣
 ↪ci, n_boot, seed, sort, err_style, err_kws, legend, ax, **kwargs)
    706     return ax
    708 p._attach(ax)
--> 710 p.plot(ax, kwargs)
    711 return ax

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/seaborn/relational.py
 ↪436, in _LinePlotter.plot(self, ax, kws)
    425 """Draw the plot onto an axes, passing matplotlib kwargs."""
    427 # Draw a test plot, using the passed in kwargs. The goal here is to
    428 # honor both (a) the current state of the plot cycler and (b) the
    429 # specified kwargs on all the lines we will draw, overriding when
   (…)
    433 # gotten from the corresponding matplotlib function, and calling the
    434 # function will advance the axes property cycle.
--> 436 scout, = ax.plot([], [], **kws)
    438 orig_color = kws.pop("color", scout.get_color())
    439 orig_marker = kws.pop("marker", scout.get_marker())

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/matplotlib/axes/_axes
 ↪py:1632, in Axes.plot(self, scalex, scaley, data, *args, **kwargs)
   1390 """
   1391 Plot y versus x as lines and/or markers.
   1392
   (…)
   1629 (``'green'``) or hex strings (``'#008000'``).
   1630 """
   1631 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1632 lines = [*self._get_lines(*args, data=data, **kwargs)]
   1633 for line in lines:
   1634     self.add_line(line)

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/matplotlib/axes/_base
 ↪py:312, in _process_plot_var_args.__call__(self, data, *args, **kwargs)
    310     this += args[0],
    311     args = args[1:]
--> 312 yield from self._plot_args(this, kwargs)
```

```
File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/matplotlib/axes/_base
 ↪py:538, in _process_plot_var_args._plot_args(self, tup, kwargs, return_kwargs
    536     return list(result)
    537 else:
--> 538     return [l[0] for l in result]

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/matplotlib/axes/_base
 ↪py:538, in <listcomp>(.0)
    536     return list(result)
    537 else:
--> 538     return [l[0] for l in result]

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/matplotlib/axes/_base
 ↪py:531, in <genexpr>(.0)
    528 else:
    529     labels = [label] * n_datasets
--> 531 result = (make_artist(x[:, j % ncx], y[:, j % ncy], kw,
    532                       {**kwargs, 'label': label})
    533           for j, label in enumerate(labels))
    535 if return_kwargs:
    536     return list(result)

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/matplotlib/axes/_base
 ↪py:351, in _process_plot_var_args._makeline(self, x, y, kw, kwargs)
    349 default_dict = self._getdefaults(set(), kw)
    350 self._setdefaults(default_dict, kw)
--> 351 seg = mlines.Line2D(x, y, **kw)
    352 return seg, kw

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/matplotlib/lines.py:
 ↪393, in Line2D.__init__(self, xdata, ydata, linewidth, linestyle, color,␣
 ↪marker, markersize, markeredgewidth, markeredgecolor, markerfacecolor,␣
 ↪markerfacecoloralt, fillstyle, antialiased, dash_capstyle, solid_capstyle,␣
 ↪dash_joinstyle, solid_joinstyle, pickradius, drawstyle, markevery, **kwargs)
    389 self.set_markeredgewidth(markeredgewidth)
    391 # update kwargs before updating data to give the caller a
    392 # chance to init axes (and hence unit support)
--> 393 self.update(kwargs)
    394 self.pickradius = pickradius
    395 self.ind_offset = 0

File ~/opt/anaconda3/envs/CDC/lib/python3.9/site-packages/matplotlib/artist.py:
 ↪1064, in Artist.update(self, props)
   1062             func = getattr(self, f"set_{k}", None)
   1063             if not callable(func):
-> 1064                 raise AttributeError(f"{type(self).__name__!r} object "
   1065                                      f"has no property {k!r}")
   1066             ret.append(func(v))
   1067 if ret:

AttributeError: 'Line2D' object has no property 'col'
```

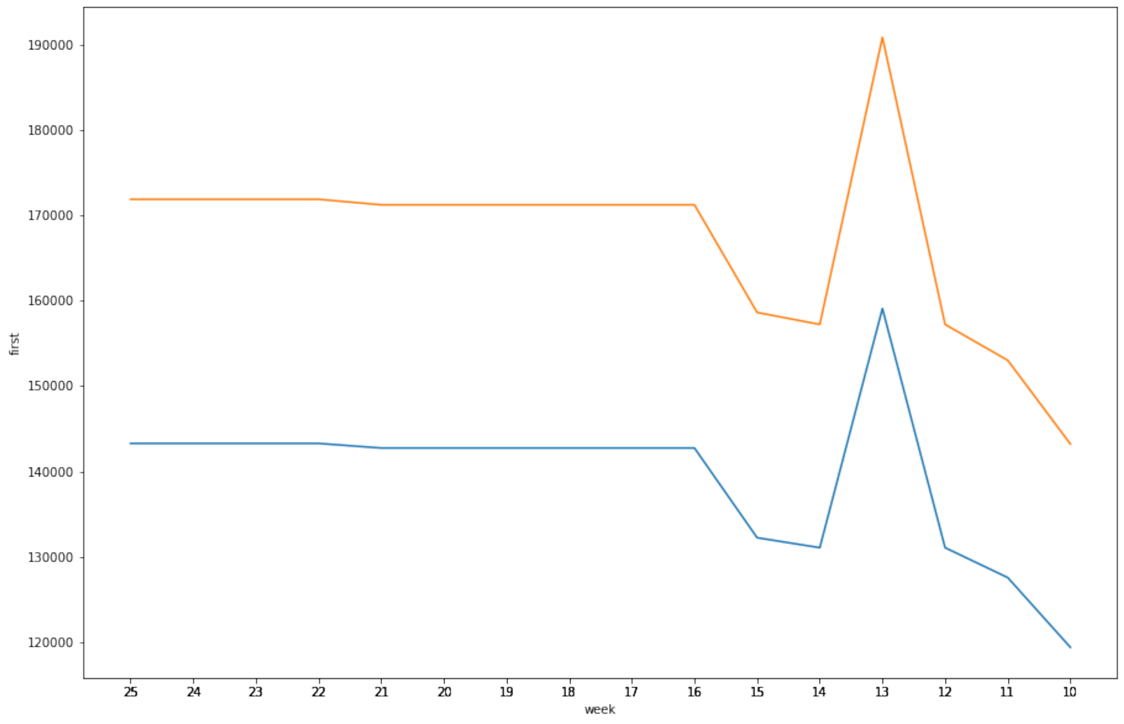


[ ]:
```
plt.figure(figsize=(12,8))
sns.relplot(data=vaccines, x='week', y='first', ci = None, estimator = 'sum',
            hue = 'jurisdiction',col='month', kind = 'line',
           col_wrap = 2)
plt.xticks(vaccines.week)
plt.show()
```

[54]:
```
# Change month to week
plt.figure(figsize=(12,8))
sns.lineplot(data=vaccines, x='week', y='first', ci = None, estimator = 'sum')
plt.xticks(vaccines.week)
plt.show()
```



[55]:
```
fig, ax1 = plt.subplots(figsize=(15, 10))
sns.lineplot(data=vaccines, x='week', y='first', ci = None, estimator = 'sum')
sns.lineplot(data=vaccines, x='week', y='second', ci = None, estimator = 'sum')
plt.xticks(vaccines.week)
plt.show()
```



[56]:
```
vaccines.columns
```

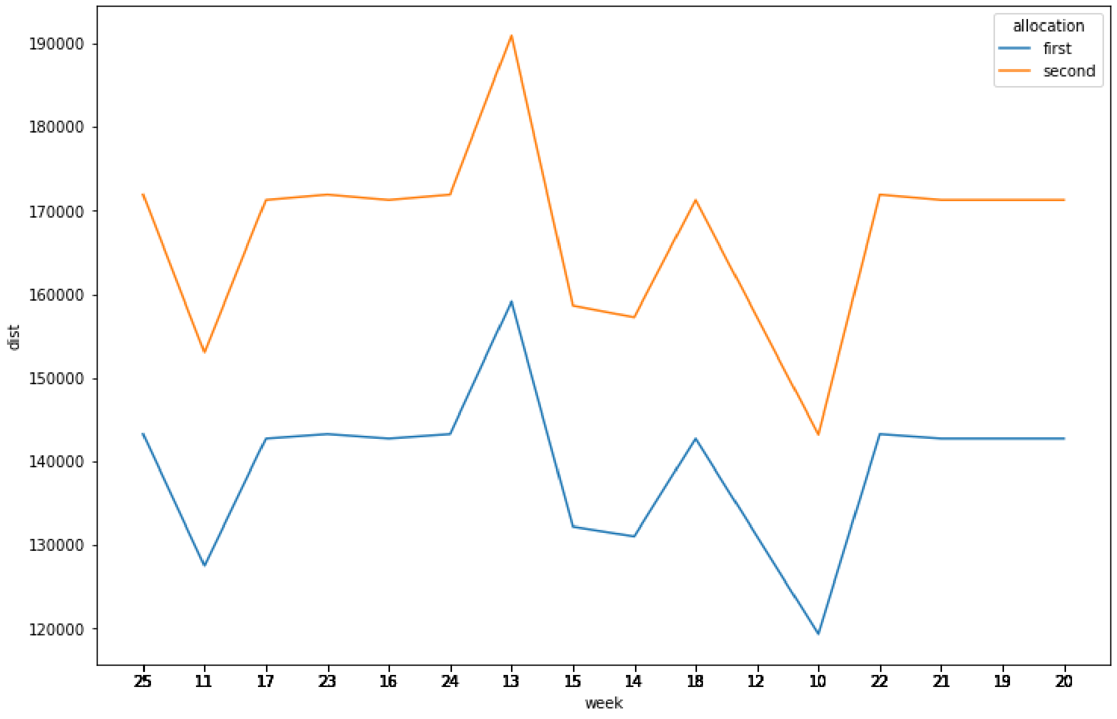
[56]:
```
Index(['jurisdiction', 'week_of_allocations', 'first', 'second', 'month',
       'week'],
      dtype='object')
```

[57]:
```
v_long = vaccines.melt(id_vars=['jurisdiction',␣
 ↪'week_of_allocations','month','week'],
                        var_name = 'allocation', value_name='dist').
 ↪sort_values(by = 'jurisdiction')
v_long.head(20)
```

[57]:
```
     jurisdiction      week_of_allocations month  week allocation      dist
0   Massachusetts  2021-06-21T00:00:00.000     6    25      first  104580.0
42  Massachusetts  2021-03-15T00:00:00.000     3    11      first   93600.0
24  Massachusetts  2021-04-26T00:00:00.000     4    17      first  105300.0
54  Massachusetts  2021-06-07T00:00:00.000     6    23     second  125496.0
27  Massachusetts  2021-04-19T00:00:00.000     4    16      first  105300.0
51  Massachusetts  2021-06-14T00:00:00.000     6    24     second  125496.0
84  Massachusetts  2021-03-29T00:00:00.000     3    13     second  140400.0
30  Massachusetts  2021-04-12T00:00:00.000     4    15      first   97110.0
78  Massachusetts  2021-04-12T00:00:00.000     4    15     second  116532.0
33  Massachusetts  2021-04-05T00:00:00.000     4    14      first   95940.0
48  Massachusetts  2021-06-21T00:00:00.000     6    25     second  125496.0
```

```
69  Massachusetts  2021-05-03T00:00:00.000     5   18     second  126360.0
36  Massachusetts  2021-03-29T00:00:00.000     3   13      first  117000.0
81  Massachusetts  2021-04-05T00:00:00.000     4   14     second  115128.0
39  Massachusetts  2021-03-22T00:00:00.000     3   12      first   95940.0
45  Massachusetts  2021-03-08T00:00:00.000     3   10      first   87750.0
87  Massachusetts  2021-03-22T00:00:00.000     3   12     second  115128.0
57  Massachusetts  2021-05-31T00:00:00.000     5   22     second  125496.0
21  Massachusetts  2021-05-03T00:00:00.000     5   18      first  105300.0
12  Massachusetts  2021-05-24T00:00:00.000     5   21      first  105300.0
```

[58]:
```
plt.figure(figsize=(12,8))
sns.lineplot(data=v_long, x='week', y='dist', ci = None, estimator = 'sum', hue␣
 ↪= 'allocation')
plt.xticks(v_long.week)
plt.show()
```



[ ]:
```
```