Group 9

NAMEs = "Atef Alqashqish", "Cameron Milligan", "Layne Moran"

COLLABORATORS = ""

In [1]:
```
# all imports and env variables
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import datetime as dt
import requests
import io
import statsmodels.api as sm
## TYPE YOUR CODE HERE
#import warnings library and ignore warning.
import warnings
warnings.filterwarnings("ignore")
```

```
/Users/cammilligan/anaconda/lib/python3.6/site-packages/statsmodels/com
pat/pandas.py:56: FutureWarning: The pandas.core.datetools module is de
precated and will be removed in a future version. Please use the panda
s.tseries module instead.
  from pandas.core import datetools
```

The first step is to retreive all of our dataframes from opendata.toronto.ca.

In [2]:
```
download_2016 = 'http://opendata.toronto.ca/it/com/ob_report_2016.xls'
download_2017 = 'http://opendata.toronto.ca/it/com/ob_report_2017.xls'
download_2018 = 'http://opendata.toronto.ca/it/com/ob_report_2018.xls'
download_2019 = 'http://opendata.toronto.ca/it/com/ob_report_2019.xls'
ob_2016 = pd.read_excel(download_2016)
ob_2017 = pd.read_excel(download_2017)
ob_2018 = pd.read_excel(download_2018)
ob_2019 = pd.read_excel(download_2019)
```

Initial Data Exploration And Data Preperation

First we check the shape of each dataframe to see a total number of rows if we combined them all

In [3]:
```
rows_count = ob_2016.shape[0] + ob_2017.shape[0] + ob_2018.shape[0] + ob
_2019.shape[0]
rows_count
```

Out[3]:
```
1329
```

Next we concat the rows together and verify that our rowcount has stayed the same.

```
In [4]: ob = pd.concat([ob_2019, ob_2018, ob_2017, ob_2016], ignore_index=True,
        sort=False)
        ob.shape[0]

Out[4]: 1329
```

Lets take a look at the first two rows to get an idea of what the data looks like.

```
In [5]: ob.head(2)
```

Out[5]:

	Institution Name	Institution Address	Outbreak Setting	Type of Outbreak	Causative Agent 1	Causative Agent 2	Date Outbreak Began	Dat Declare Ov
0	Greenview Lodge	880 Lawrence Ave E	Retirement Home	Respiratory	Pending	NaN	2019-09-07	NaT
1	North York General Hospital - Seniors' Health ...	2 Buchan Crt	LTCH	Respiratory	Pending	NaN	2019-08-28	NaT

Lets also call .info() to see each columns data types and count of nulls.

```
In [6]: ob.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1329 entries, 0 to 1328
Data columns (total 13 columns):
Institution Name       1329 non-null object
Institution Address    1329 non-null object
Outbreak Setting       1329 non-null object
Type of Outbreak       1329 non-null object
Causative Agent 1      251 non-null object
Causative Agent 2      17 non-null object
Date Outbreak Began    1329 non-null datetime64[ns]
Date Declared Over     1325 non-null datetime64[ns]
Active                 1329 non-null object
Causative Agent-1      810 non-null object
Causative Agent-2      109 non-null object
Causative Agent        268 non-null object
Etiological Agent2     33 non-null object
dtypes: datetime64[ns](2), object(11)
memory usage: 135.1+ KB
```

Data Preparation of Caustive Agent

In order to simplify some analysis, we want to merge the three Caustive Agent columns into one column. We do this with a lambda function to simply join the data together.

In [7]:
```
#Merge Causative Agent into one column named disease and all Agent colum
ns
ob['disease'] = ob.loc[:, ("Causative Agent-1", "Causative Agent-2", "Ca
usative Agent")].apply(lambda x: " ".join(x.astype(str)), axis=1)
ob['disease'] = ob['disease'].str.replace("nan", "")
```

We know we will have white spaces, so we will strip them out. We will also drop the three Causative Agent columns because we have merged them into our new diesease column.

In [8]:
```
ob['disease'] = ob['disease'].str.strip(' ')
ob.drop(["Causative Agent-1", "Causative Agent-2", "Causative Agent"],ax
is=1, inplace=True)
ob.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1329 entries, 0 to 1328
Data columns (total 11 columns):
Institution Name       1329 non-null object
Institution Address    1329 non-null object
Outbreak Setting       1329 non-null object
Type of Outbreak       1329 non-null object
Causative Agent 1      251 non-null object
Causative Agent 2      17 non-null object
Date Outbreak Began    1329 non-null datetime64[ns]
Date Declared Over     1325 non-null datetime64[ns]
Active                 1329 non-null object
Etiological Agent2     33 non-null object
disease                1329 non-null object
dtypes: datetime64[ns](2), object(9)
memory usage: 114.3+ KB
```

Now, as you can see, disease column which holds the diseases names has no NaN values. Yet, we need to do clean up and make sure the diseases are classified well without repetition

Our Etiological Agent2 column only has 33 non-null values. This is not enough to conduct an analysis on, so we will drop this column as well

In [9]:
```
ob.drop(["Etiological Agent2"],axis=1, inplace=True)
```

```
In [10]: #Check what data do we have under Causative Agent All
         ob['disease'].value_counts().head(20)
```

```
Out[10]:                                                  251
         Unable to identify                               203
         Influenza A (H3)                                 147
         Norovirus-like                                   107
         Influenza B                                      105
         Rhinovirus                                        72
         Respiratory syncytial virus                       67
         Influenza A (H3N2)                                40
         Influenza A (Not subtyped)                        40
         Enterovirus/Rhinovirus                            37
         Coronavirus                                       34
         Influenza A ((H1N1)pdm09)                         32
         Metapneumovirus                                   31
         Parainfluenza type 3                              26
         Influenza A (H3) Respiratory syncytial virus      16
         Influenza A (H3) Influenza B                      11
         Influenza A (H3) Coronavirus                      10
         Influenza B Respiratory syncytial virus            9
         Clostridium difficile                              7
         Parainfluenza type 1                               4
         Name: disease, dtype: int64
```

In [11]:
```
#Les't clean up disease column more to group disease and identify the ty
pe
#convert the string to lower case
ob['disease'] = ob['disease'].str.lower()
ob.head()
```

Out[11]:

	Institution Name	Institution Address	Outbreak Setting	Type of Outbreak	Causative Agent 1	Causative Agent 2	Date Outbreak Began	Da Declar Ov
0	Greenview Lodge	880 Lawrence Ave E	Retirement Home	Respiratory	Pending	NaN	2019-09-07	NaT
1	North York General Hospital - Seniors' Health ...	2 Buchan Crt	LTCH	Respiratory	Pending	NaN	2019-08-28	NaT
2	Hellenic Home - 2 A Wing	2411 Lawrence Ave E	LTCH	Respiratory	Unable to identify	NaN	2019-08-28	2019-0 05
3	Michael Garron Hospital - 2nd Fl - ICU	825 Coxwell Ave	Hospital-Acute Care	Enteric	Clostridium difficile	NaN	2019-08-27	NaT
4	The Wexford - 4th Fl	1860 Lawrence Ave E	LTCH	Respiratory	Unable to identify	NaN	2019-08-26	2019-0 09

In [12]:
```
def extract_disease_name(disease):
    '''
    extract_disease_name(str) -> panads Series (disease_name, disease_ty
pe, disease_sub_type)

    THe funciton takes a string which a string descripe the disease and
 extract disease_name,
    disease_type, disease_sub_tye based on specific conditions, check th
e if statements
    '''
    disease_str = str(disease) #first make sure the parameter is string
    disease_str = disease_str.split(' ') #split the string into a list b
ased on a space
    #append nan in the array to avoid errors, we expect an array of min
 4 values
    for i in range(len(disease_str), 4):
        disease_str.append(str(np.nan))
    #disease string equal 'unable to identify' or 'pending', do nothing
    if (disease == 'unable to identify' or disease == 'pending'):
        return pd.Series((disease, np.nan, np.nan))
    #case when disease string starts with 'influenza', usualy it is foll
owed by type A or B
    if (disease_str[0] == 'influenza'):
        return pd.Series((disease_str[0], disease_str[1], ' '.join(disea
se_str[2:])))
    #case when disease string has 'type' word as the second word
    elif (disease_str[1] == 'type'):
        return pd.Series((disease_str[0], disease_str[2], ' '.join(disea
se_str[3:])))
    #otherwise, return first word as the disease name, NaN as the type a
nd the reset as the sub_type
    else:
        return pd.Series((disease_str[0], np.nan, ' '.join(disease_str[1
:])))

#Apply the funciton to the DataFrame
ob[['d_name', 'd_type', 'd_sub_type']] = ob['disease'].apply(extract_dis
ease_name)
#Some d_sub_type values will contain nan string, let's clean them up
ob['d_sub_type'] = ob['d_sub_type'].str.replace('nan', '')
ob['d_sub_type'].replace('', np.nan, inplace=True)
#remove white spaces at the start and the end of strings
ob['d_sub_type'] = ob['d_sub_type'].str.strip(' ')
#ob['disease_type'] = ob['disease'].apply(extract_disease_type, disease_
name= ob['disease_name'])
ob.head(2)
```

Out[12]:

	Institution Name	Institution Address	Outbreak Setting	Type of Outbreak	Causative Agent 1	Causative Agent 2	Date Outbreak Began	Dat Declare Ov
0	Greenview Lodge	880 Lawrence Ave E	Retirement Home	Respiratory	Pending	NaN	2019-09-07	NaT
1	North York General Hospital - Seniors' Health ...	2 Buchan Crt	LTCH	Respiratory	Pending	NaN	2019-08-28	NaT

Data Preparation of Institution Name

First we will rename a couple of our columns so theya re easier to work with.

In [13]:
```
ob.rename(columns = {'Institution Name':'Institution_Name','Institution
 Address':'Institution_Address'}, inplace = True)
ob.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1329 entries, 0 to 1328
Data columns (total 13 columns):
Institution_Name       1329 non-null object
Institution_Address    1329 non-null object
Outbreak Setting       1329 non-null object
Type of Outbreak       1329 non-null object
Causative Agent 1      251 non-null object
Causative Agent 2      17 non-null object
Date Outbreak Began    1329 non-null datetime64[ns]
Date Declared Over     1325 non-null datetime64[ns]
Active                 1329 non-null object
disease                1329 non-null object
d_name                 1329 non-null object
d_type                 489 non-null object
d_sub_type             1090 non-null object
dtypes: datetime64[ns](2), object(11)
memory usage: 135.1+ KB
```

Our objective is to clean the Institution Name column. Currently the column has additional data that we are not interested in such as floor name or hospital wing. For example we want to clean values such as "Sunnybrook Health Sciences Centre - C4" to become ""Sunnybrook Health Sciences Centre".

These additional details always occur after a "-" so we wrote a function which splits on the special character "-" and strips everything after it.

In [14]:
```
def delete_character(word):
    return word.split(' -')[0].split('- ')[0].strip()
```

We then apply this function on the Institution Name and Institution Address fields.

In [15]:
```
ob['Institution_Name'] = ob['Institution_Name'].apply(delete_character).
str.title()
ob['Institution_Address'] = ob['Institution_Address'].apply(delete_chara
cter).str.title()
```

Here is taken as a reference the Institution Address and is replaced by the Institution Name for the most frequent Institution Name per group, that will allow us to correctly determine the diseases and duration by Institution.

We then create a new dataframe which groups the data by the cleaned Institution_address. We then loop over our original dataframe and replace the institution address with the most frequent value based on the group by.

The end result is a cleaned Institution_Name column with the additional details after the dash stripped out.

In [16]:
```
ob1 = ob.groupby(["Institution_Address"], as_index=False)["Institution_N
ame"].first()


for idx, record in ob['Institution_Address'].iteritems():
    for i in ob1.values:
            if (i[0] == record):
                ob.loc[ob.Institution_Address == record, 'Institution_Na
me'] = i[1]
```

To visually see the result we use Sunnybrook as an example. We slice our cleaned ob dataframe and our backup_ob dataframe by the same filter (Sunybrook Health Science Centre) in order to return the same rows. We can see that our cleaned dataframe has stripped out the extra values.

In [17]:
```
ob.loc[ob['Institution_Name'] == 'Sunnybrook Health Sciences Centre'].he
ad(2)
```

Out[17]:

	Institution_Name	Institution_Address	Outbreak Setting	Type of Outbreak	Causative Agent 1	Causative Agent 2
5	Sunnybrook Health Sciences Centre	2075 Bayview Ave	Hospital-Acute Care	Enteric	Clostridium difficile	NaN
8	Sunnybrook Health Sciences Centre	200 Church St	Hospital-Chronic Care	Respiratory	Parainfluenza type 3	NaN

Data Preparation of Duration

We have a Date Outbreak Began column and a Date Declared Over column. We can use these two columns to calculate an Outbreak Duration. Before we can calculate this we need to deal with our NaT values. We have NaT values when the disease outbreak is still active. To handle this case we replace the NaT with the current date.

```
In [18]: ob['Date Declared Over'][ob['Date Declared Over'].isna() == True ] = pd.
         to_datetime(dt.datetime.now().date())
         #Create a duration column with a number of days the outbreak was active
         ob['Duration'] = ob['Date Declared Over'] - ob['Date Outbreak Began']
```

Analysis

The Most Prevalent Diseases Across the Data Set

```
In [19]: #check the most prevalent diseases across the Data Set
         ob['d_name'].value_counts().sort_values(ascending = False).head()
```

```
Out[19]: influenza             450
                               251
         unable to identify    203
         norovirus-like        107
         rhinovirus             74
         Name: d_name, dtype: int64
```

Result1

The most prevalent diseases across the Data Set is: Influenza followed by Parainfluenza

```
In [20]: #Let's check most prevalent diseases across the Data Set based on name a
         nd type
         ob.groupby(['d_name', 'd_type'])['disease'].count().sort_values(ascendin
         g = False).head()
```

```
Out[20]: d_name         d_type
         influenza      a         328
                        b         122
         parainfluenza  3          28
                        1           5
                        4           4
         Name: disease, dtype: int64
```

Result 2

The most prevalent diseases across the Data Set is: Influenza Type A followed by Influenza Type B

What are the most prevalent diseases in each of the outbreak settings

In [21]:
```
#Let's first clean up Outbreak Setting into two columns: ob_st_name and
 ob_st_sub_name
# When exploring the Outbreak Setting column we noticed that Hospitals h
ad a sub setting.
# For example, the value "Hospital-Acute Care" should be divided into "H
ospital" with a subtype of "Acute Care"
# This way we can group all of the Hospitals as one category for a broad
er analysis.

def extract_outbreak_setting_name(outbreak):
    '''
    extract_outbreak_setting_name(str) -> panads Series (ob_st_name, ob_
st_sub_name)

    The funciton takes a string which a string describe the outbreak set
ting and
    outbreak setting name (ob_st_name) and outbreak setting subname (ob_
st_sub_name)
    based on specific conditions, check the if statements
    '''
    outbreak_str = str(outbreak) #first make sure the parameter is strin
g
    outbreake_str = outbreak_str.split('-') #split the string into a lis
t based on a dash
    #append nan in the array to avoid errors, we expect an array of min
 3 values to avoid errors
    for i in range(len(outbreake_str), 3):
        outbreake_str.append(str(np.nan))
    return pd.Series((outbreake_str[0], outbreake_str[1]))


#Apply the funciton to the DataFrame
ob[['ob_st_name', 'ob_st_sub_name']] = ob['Outbreak Setting'].apply(extr
act_outbreak_setting_name)
ob.head(2)
```

Out[21]:

	Institution_Name	Institution_Address	Outbreak Setting	Type of Outbreak	Causative Agent 1	Causative Agent 2	O
0	Greenview Lodge	880 Lawrence Ave E	Retirement Home	Respiratory	Pending	NaN	20 07
1	North York General Hospital	2 Buchan Crt	LTCH	Respiratory	Pending	NaN	20 28

In [22]:
```
#check most prevalent diseases across the Data Set per Outbreak Setting,
disease name and type
disease_outbreak_setting = ob.groupby(['ob_st_name', 'd_name'])['diseas
e'].count().sort_values(ascending = False).to_frame()
disease_outbreak_setting.head()
```

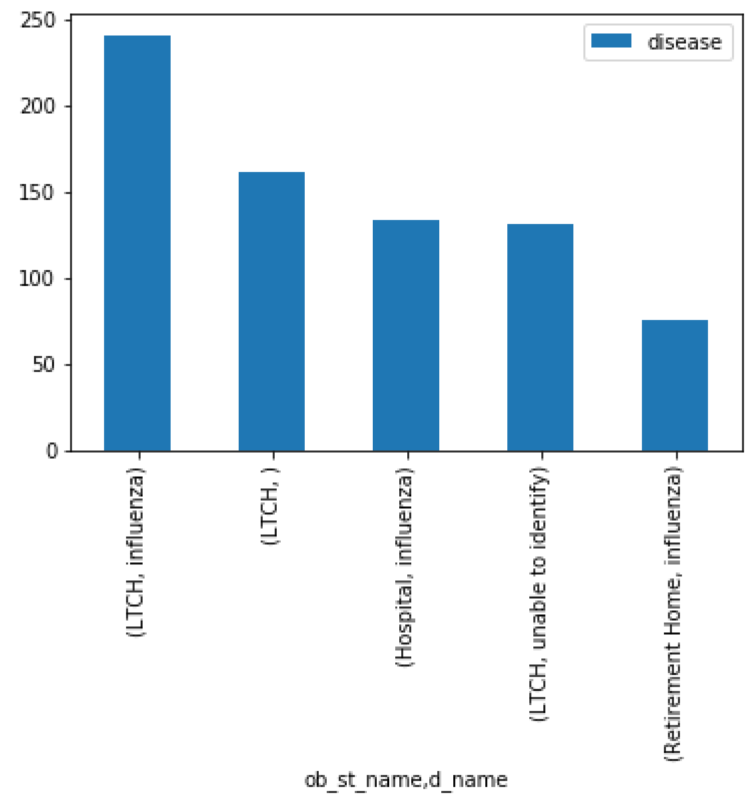
Out[22]:

		disease
ob_st_name	**d_name**	
LTCH	**influenza**	241
		161
Hospital	**influenza**	133
LTCH	**unable to identify**	131
Retirement Home	**influenza**	75

In [23]:
```
ob.groupby(by="Outbreak Setting")['Duration'].sum()
ob.groupby(by="ob_st_name")['Duration'].sum()
```

Out[23]:
```
ob_st_name
Hospital            3301 days
LTCH               12003 days
Retirement Home     2783 days
Shelter              650 days
Name: Duration, dtype: timedelta64[ns]
```

In [24]:
```
disease_outbreak_setting.head().plot(kind='bar')
None
```



Which diseases occurs most frequently

In [25]:
```
#Extract disease names in an array
d_names = ob[ob['d_name'] != 'pending' ]['d_name'].to_frame()
d_names = d_names [ d_names ['d_name'] != 'unable to identify']
d_names = d_names['d_name'].unique()
```

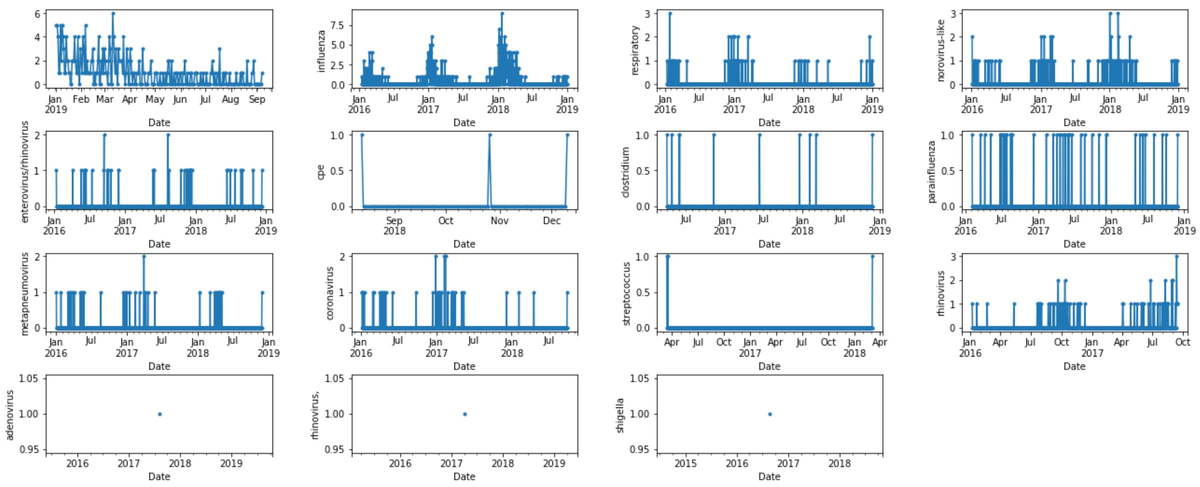
In [26]:
```
#set the index to be a time series
ob.index = ob['Date Outbreak Began']
```

In [27]:
```
def diseases_freq_and_plot(ob, d_names, freq):
    '''
    diseases_freq_and_plot(DataFrame, Array, Freqeuncy) -> DataFrame, In
teger

    The function resample the data set using Frequency variable, plot ea
ch disease outbreak over time and
    caculate the total number of frequency in the dataset period

    '''
    #count how many frequency there are in the data set period, required
to calculate the average of outbreak
    # during the dataset period
    start = min(ob['Date Outbreak Began'])
    end = max (ob['Date Outbreak Began'])
    freq_count = pd.period_range(start = start, end = end, freq = freq)
    freq_count = len(freq_count)

    #Data Frame to collect number of outbreak per disease per frequency.
We assume a disease outbreaks frequently if it
    # indeed outbreak at least once pre frequency
    outbreaks_frequency = pd.DataFrame(columns=['date', 'd_name', 'freq'
])

    #plot how many each disease outbreak each month
    fig = plt.figure(figsize=(20,10))
    a = 0
    for i in range(4):
        if (a >= len(d_names)):
            break
        for j in range(4):
            if (a >= len(d_names)):
                break
            d_name = d_names[a]
            ax = plt.subplot2grid((5,4), (i,j))
            most_frequent = ob['d_name'][ob['d_name'] == d_name].resampl
e(freq).count()
            for d_count in range(len(most_frequent)):
                outbreaks_frequency.loc[len(outbreaks_frequency)] = [mos
t_frequent.index.values[d_count],
                                                                   d_name,
                                                                  most_fre
quent[d_count]]
            m_plot = most_frequent.plot(kind='line', style='.-')
            m_plot.set_ylabel(d_name)
            m_plot.set_xlabel('Date')
            a = a + 1
    plt.subplots_adjust(top=0.92, bottom=0.08, left=0.10, right=0.95, hs
pace=0.55,
                        wspace=0.35)
    plt.show()
    return outbreaks_frequency, freq_count
```

In order to answer the question "Which diseases occurred most frequently", we will use this formula:

The diseases occurred most frequently = The diseases with the largest number of outbreaks per

```
In [28]: #let's start by seek the top 3 diseases occured most frequely using Days
         as the frequenct
         freq = 'd'
         outbreaks_frequency, freq_count = diseases_freq_and_plot(ob, d_names, fr
         eq)
```



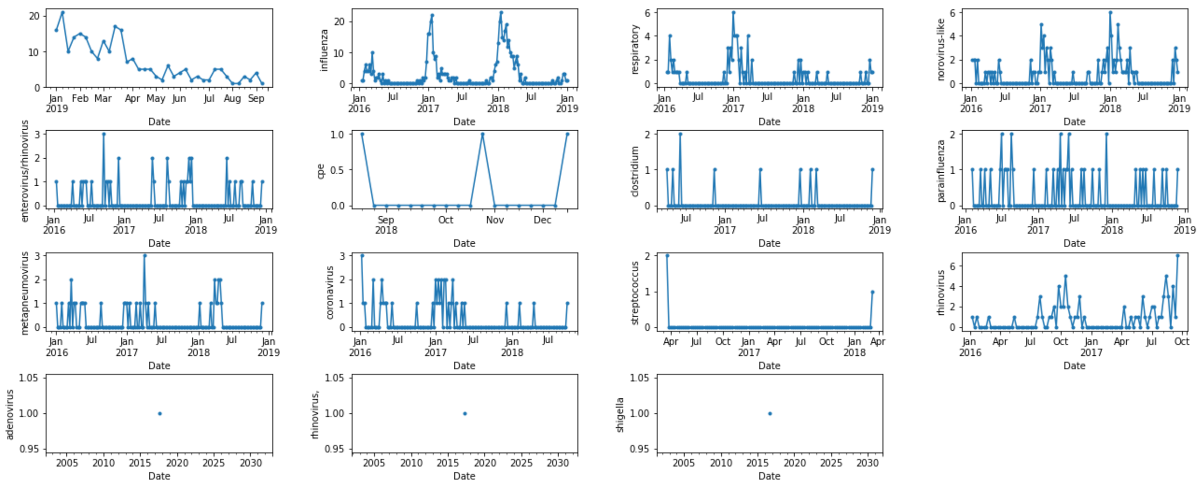
```
In [29]: def caculate_disease_avg_frequency(outbreaks_frequency, freq_count, d_na
         mes):
             '''
             caculate_disease_avg_frequency(DataFrame, Integer, Array) -> DataFra
         me (contains: diseas name as string,
             number of occurance as an integer, the total number of frequency in
          the dataset period as integer)

             The funciton caclulate the number of occurance per disease and retur
         n the result in a DataFrame format

             '''
             result = pd.DataFrame(columns=['d_name', 'disease_avg_frequency', 'f
         req_count'])
             for d_name in d_names:
                 one_disease = outbreaks_frequency[ outbreaks_frequency['d_name']
         == d_name]
                 one_disease = one_disease[one_disease['freq'] >= 1 ].count()
                 one_disease = one_disease['d_name']
                 result.loc[len(result)] = [d_name, one_disease, freq_count]
             return result
```

In [30]:
```
result = caculate_disease_avg_frequency(outbreaks_frequency, freq_count,
d_names)
result.index = result['d_name'].values
result = result[['disease_avg_frequency', 'freq_count']].sort_values(by=
'disease_avg_frequency', ascending = False)
result.head(3)
```

Out[30]:

	disease_avg_frequency	freq_count
influenza	237	1344
	139	1344
norovirus-like	95	1344

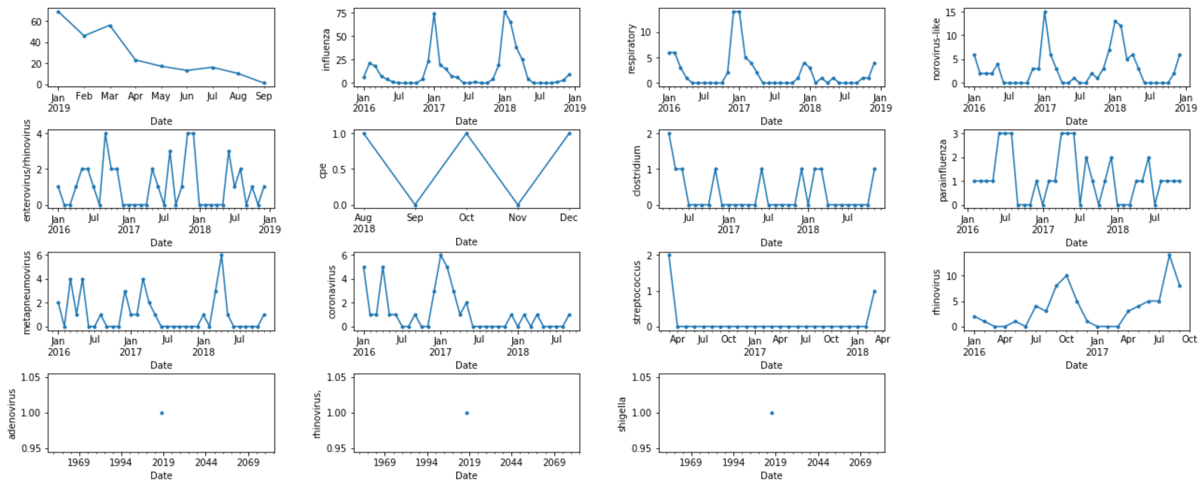
In [31]:
```
#let's examin the top 3 diseases occured most frequely using Weeks as th
e frequenct
freq = 'W'
outbreaks_frequency, freq_count = diseases_freq_and_plot(ob, d_names, fr
eq)
result = caculate_disease_avg_frequency(outbreaks_frequency, freq_count,
d_names)
result.index = result['d_name'].values
result = result[['disease_avg_frequency', 'freq_count']].sort_values(by=
'disease_avg_frequency', ascending = False)
result.head(3)
```



Out[31]:

	disease_avg_frequency	freq_count
influenza	80	193
norovirus-like	57	193
respiratory	39	193

In [32]:
```
#let's examin the top 3 diseases occured most frequely using Months as t
he frequenct
freq = 'M'
outbreaks_frequency, freq_count = diseases_freq_and_plot(ob, d_names, fr
eq)
result = caculate_disease_avg_frequency(outbreaks_frequency, freq_count,
d_names)
result.index = result['d_name'].values
result = result[['disease_avg_frequency', 'freq_count']].sort_values(by=
'disease_avg_frequency', ascending = False)
result.head(3)
```



Out[32]:

	disease_avg_frequency	freq_count
influenza	24	45
parainfluenza	24	45
norovirus-like	22	45

Which disease had the longest durations

First we have to set the index back to a number count. Then we can

In [33]:
```
ob.index = range(len(ob.index))
```

In [34]:
```
ob[ob['Duration'] == max(ob['Duration'])]['d_name'].values
```

```
Out[34]: array(['streptococcus'], dtype=object)
```

The longest outbreak was a streptococcus outbreak that lasted 578 days in a Shelter.

```
In [35]: #The longest outbreak. 578 days!
         ob.loc[ob['Duration'].idxmax()]
```

```
Out[35]: Institution_Name                    Seaton House
         Institution_Address                339 George St
         Outbreak Setting                         Shelter
         Type of Outbreak                     Respiratory
         Causative Agent 1                            NaN
         Causative Agent 2                            NaN
         Date Outbreak Began          2016-03-17 00:00:00
         Date Declared Over           2017-10-16 00:00:00
         Active                                         N
         disease                  streptococcus pyogenes
         d_name                             streptococcus
         d_type                                       NaN
         d_sub_type                              pyogenes
         Duration                     578 days 00:00:00
         ob_st_name                               Shelter
         ob_st_sub_name                               nan
         Name: 1237, dtype: object
```

We can see that the average outbreak is 14 days with 75% of the data falling under 16 days. This indicates that the 578 day outbreak was a significant outlier.

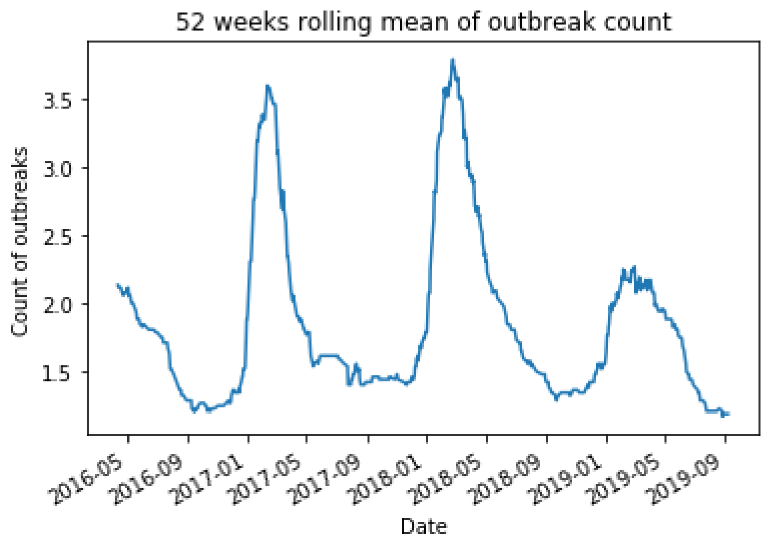
```
In [36]: ob['Duration'].describe()
```

```
Out[36]: count                       1329
         mean     14 days 02:21:56.478555
         std      17 days 13:10:47.277743
         min              2 days 00:00:00
         25%              9 days 00:00:00
         50%             12 days 00:00:00
         75%             16 days 00:00:00
         max            578 days 00:00:00
         Name: Duration, dtype: object
```

What is the seasonality trend of the data?

The following two charts examine the seasonality of the outbreak dataset. The 52 week rolling mean smooths outliers and allows a strong visualisation of the seasonality. It is clear that in the winter months is a spike in the number of outbreaks that begin.

In [37]:
```
timeseries_ob = ob.set_index('Date Outbreak Began')
timeseries_ob_forplotting = timeseries_ob.groupby('Date Outbreak Began')
['Type of Outbreak'].count()
ob_mean = timeseries_ob_forplotting.rolling(52).mean()
#ob_mean = ob_mean.fillna(method='bfill')
ax = ob_mean.plot()
ax.set_xlabel("Date")
ax.set_ylabel("Count of outbreaks")
ax.set_title("52 weeks rolling mean of outbreak count")
plt.show()
```



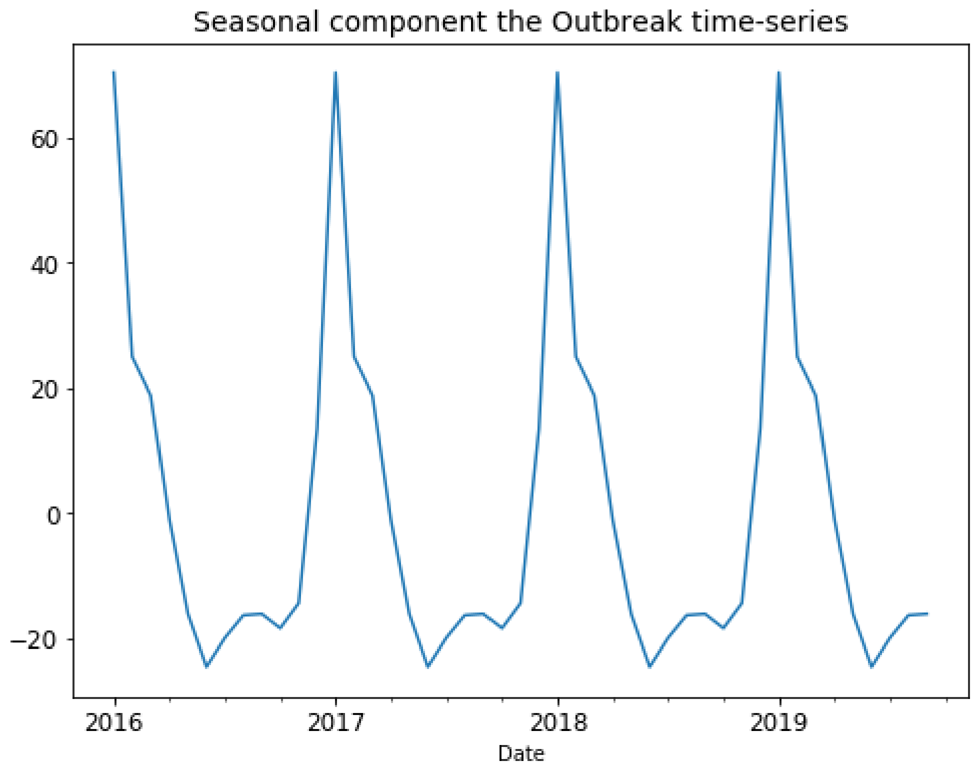
In [38]:
```
outbreak_timseries = ob.groupby('Date Outbreak Began')['disease'].count
()
```

In [39]:
```
outbreak_monthly = outbreak_timseries.resample("M").sum()
```

In [40]:
```
# Perform time series decompositon
decomposition = sm.tsa.seasonal_decompose(outbreak_monthly)
```

In [41]:
```
# Extract seasonal component
seasonal = decomposition.seasonal
```

```
In [42]: ax = decomposition.seasonal.plot(figsize=(8, 6), fontsize=12)
         ax.set_xlabel('Date', fontsize=10)
         ax.set_title('Seasonal component the Outbreak time-series', fontsize=14)
         plt.show()
```



Which month has the most outbreaks?

```
In [43]: ob.sort_values(by='Date Outbreak Began').head(2)
```

Out[43]:

	Institution_Name	Institution_Address	Outbreak Setting	Type of Outbreak	Causative Agent 1	Causative Agent 2
1328	Fairview Nursing Home	14 Cross St	LTCH	Respiratory	NaN	NaN
1327	The Heritage Nursing Home	1195 Queen St E	LTCH	Enteric	NaN	NaN

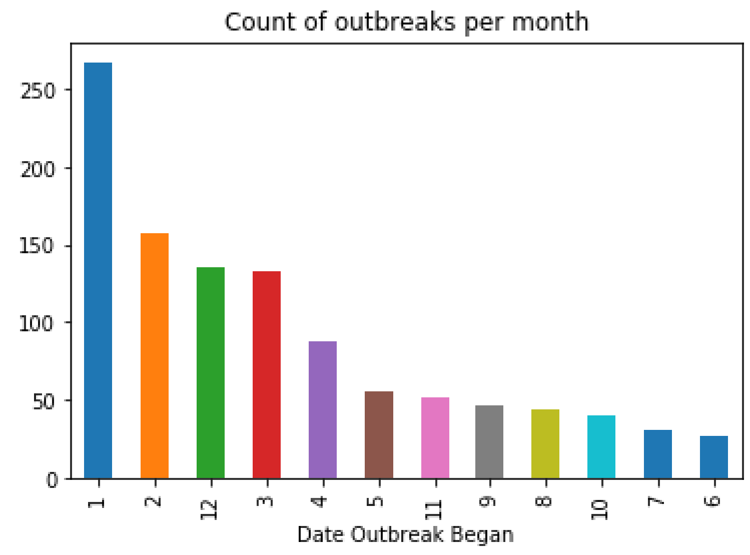
Our data begins in January 2016 and goes until March 2019. We will cut out 2019 so that the winter months are not over weighted.

```
In [44]: index_monthly = outbreak_monthly['2016':'2018'].index.month
```

```
In [45]: outbreaks_per_month = outbreak_monthly['2016':'2018'].groupby(index_mont
         hly).sum()
```

In [46]:
```
outbreaks_per_month.sort_values(ascending=False).plot(kind='bar',label=
'Count',
                                                     title='Count of ou
tbreaks per month')
```

Out[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x124a12518>
```



January, February, and December are the months with the highest overall count of outbreaks. This is consistent with the hypothesis that the winter months have the most outbreaks.

What could 2019 look like based on 2018?

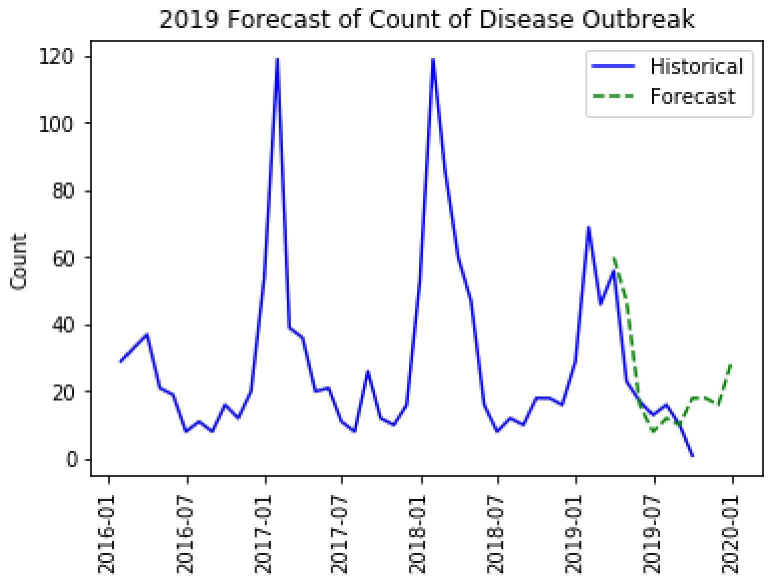
Below is a forecast that simply uses 2018 values to predict 2019s values. This is a form of forecasting called naive forecast.

In [47]:
```
ts_monthly = timeseries_ob_forplotting.resample("M").sum()
```

In [48]:
```
forecast_ts = pd.DataFrame(index=pd.date_range('2019-3-1', periods=10, f
req='M'))
```

In [49]:
```
forecast_ts['Values'] = timeseries_ob_forplotting.resample("M").sum()['2
018-03-01':'2018-12-31'].values
```

```
In [50]: plt.plot(ts_monthly,'b-' , label='actual')
         plt.plot(forecast_ts,'g--',label='forecast')
         plt.xticks(rotation='vertical')
         plt.ylabel('Count')
         plt.title('2019 Forecast of Count of Disease Outbreak')
         plt.legend(['Historical','Forecast'])
         plt.show()
```



Conclusion

- The top 3 diseases occured most frequely assuming 'daily' frequency are:
  - influenza, occured 283 times out of 1173
  - norovirus-like, occured 104 times out of 1173
  - respiratory, occured 85 times out of 1173
- The top 3 diseases occured most frequely assuming 'weekly' frequency are:
  - influenza, occured 91 times out of 169
  - norovirus-like, occured 62 times out of 169
  - respiratory, occured 48 times out of 169
- The top 3 diseases occured most frequely assuming 'monthly' frequency are:
  - influenza, occured 27 times out of 39
  - norovirus-like, occured 26 times out of 39
  - respiratory, occured 25 times out of 39

References

APA Citation Style:

Communicable Disease Surveillance Unit, Toronto Public Health. (2019). Outbreaks in Toronto Healthcare Institutions (2019) [Data file]. Retrieved from https://www.toronto.ca/city-government/data-research-maps/open-data/open-data-catalogue/health/#bfd188df-008a-e625-bd2a-e5430439dcef (https://www.toronto.ca/city-government/data-research-maps/open-data/open-data-catalogue/health/#bfd188df-008a-e625-bd2a-e5430439dcef)