In [1]:
```
! ls /Users/user/Downloads/FARS2019NationalCSV/
```

```
ACC_AUX.csv     Factor.CSV      NMDistract.CSV Person.CSV      Vehicle.CSV
CEvents.CSV     MIACC.csv       NMImpair.CSV    Race.CSV        Violatn.CSV
Damage.CSV      MIDRVACC.csv    NMPrior.CSV     SafetyEq.CSV    Vision.CSV
Distract.CSV    MIPER.csv       PBType.CSV      VEH_AUX.csv     accident.CS
V
DrImpair.CSV    Maneuver.CSV    PER_AUX.csv     VEvent.CSV
Drugs.CSV       NMCrash.CSV     Parkwork.CSV    VSOE.CSV
```

In [79]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
import glob
from tqdm.notebook import tqdm
import seaborn as sns
sns.set()
from datetime import datetime
import matplotlib.pyplot as plt
```

In [3]:
```python
import importlib
# importlib.reload(sns)
```

In [4]:
```python
gsa_codes = pd.read_excel('/Users/user/Downloads/FRPP_GLC_-_United_State
sDEC72020.xlsx')
gsa_codes = gsa_codes.rename(columns={'City Name':'CITY NAME', 'City Cod
e':'CITY', 'State Name':'STATE NAME', 'State Code':'STATE'})

# let's look at the info for the cites we want to compare
gsa_codes[((gsa_codes['CITY NAME']=='NEW YORK')&(gsa_codes['STATE NAME']
=='NEW YORK'))
          |((gsa_codes['CITY NAME']=='DETROIT')&(gsa_codes['STATE NAME']
=='MICHIGAN'))]
```

Out[4]:

	Territory	STATE NAME	STATE	CITY	CITY NAME	County Code	County Name	Country Code	Old City Name	Date Record Added	D
15948	U	MICHIGAN	26	1260	DETROIT	163	WAYNE	840	NaN	NaT	2
23761	U	NEW YORK	36	4170	NEW YORK	61	NEW YORK	840	NEW YORK CITY	NaT	3

In [5]:
```python
# let's load up all of the tables we plan to use
DFdict = {
    'accident':None,
    'Person': None
}
# grab the tables we want and concatenate them across years
for key in tqdm(DFdict.keys()):
    dflist = []
    fids = glob.glob(f'/Users/user/Downloads/*/{key}.CSV')
    for fid in tqdm(fids, leave=False):
        dflist.append(pd.read_csv(fid, encoding="ISO-8859-1", engine='py
thon'))
    DFdict[key] = pd.concat(dflist, axis=0)
```

In [6]:
```python
# we only care about DETROIT and NEW YORK right now so let's filter all
 of our data just keep those
# let's join the city and state names onto our data using the GSA datafr
ame
df_a = DFdict['accident']
df_a = df_a.merge(gsa_codes[['CITY NAME', 'CITY', 'STATE NAME', 'STATE'
]], on=['STATE', 'CITY'], how='left')
# we can also filter the DF to only include the data for the cities we c
are about
df_a = df_a[((df_a['CITY NAME']=='NEW YORK')&(df_a['STATE NAME']=='NEW Y
ORK'))
        |((df_a['CITY NAME']=='DETROIT')&(df_a['STATE NAME']=='MICHIGAN'
))]

nyc_det_cases = set(df_a.ST_CASE.dropna().unique())
```

In [111]:
```python
# we also only want to look at pedestrian cases, so we will filter for S
T_CASEs that include at least one pedestrian
# this is where PER_TYPE = 4, 5, 6, 7, 8, 10 or 19
df_p = DFdict['Person']
pedestrian_cases = set(df_p[df_p.PER_TYP.isin([4,5,6,7,8,10,19])].ST_CAS
E.dropna().unique())
```

In [112]:
```python
# we want to now just keep cases that are either in NYC or DET, and have
at least one pedestrian involved
# here we take the intersection of the cases we collected above
keep_cases = nyc_det_cases.intersection(pedestrian_cases)

# now we can filter out data to just those rows matching the selected ST
_CASEs
df_a = df_a[df_a.ST_CASE.isin(keep_cases)]
df_p = df_p[df_p.ST_CASE.isin(keep_cases)]
```

In [113]:

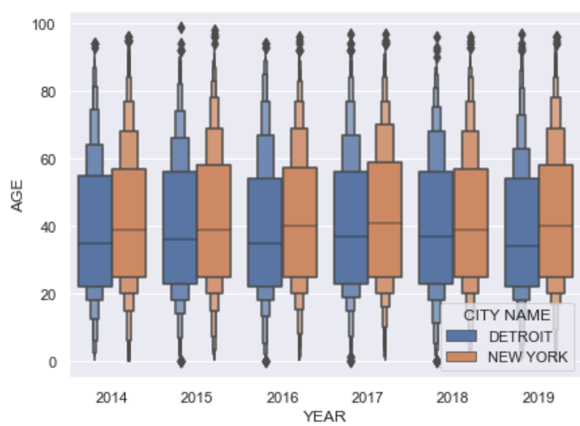
```
# let's also merge the city info into the person data, so it's easier to
look at the comparative aggregates
df_p = df_p.merge(df_a[['ST_CASE', 'CITY NAME', 'YEAR']], on='ST_CASE',
how='left')

# the documentation also indicates that for the person dataframe only 0-
120 are actual ages
df_p['AGE'] = df_p['AGE'].map(lambda x: None if x>120 else x)
```

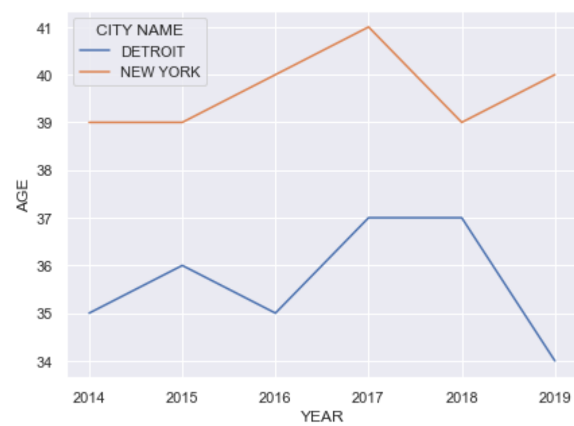
In [114]:

```
# Let's first look at whether there is a significant difference in the a
ge of people involved in fatal pedestrian crashes
# and how that may change over time
fig, axes = plt.subplots(1, 2, figsize=(15, 5))
sns.boxenplot(data=df_p, x="YEAR", y="AGE", hue="CITY NAME", ax=axes[0])

crash_ages = df_p.groupby(['CITY NAME', 'YEAR']).AGE.quantile(0.5).reset
_index()
sns.lineplot(data=crash_ages, x='YEAR', y='AGE', hue='CITY NAME', ax=axe
s[1])

# clearly the age of people involved in crashes tends to be lower in DET
ROIT than in NYC
# For future we can explore this further by looking at the breakdown of
 persons by sex, in-vehicle/pedestrian, and fatality
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd3ab692690>
```

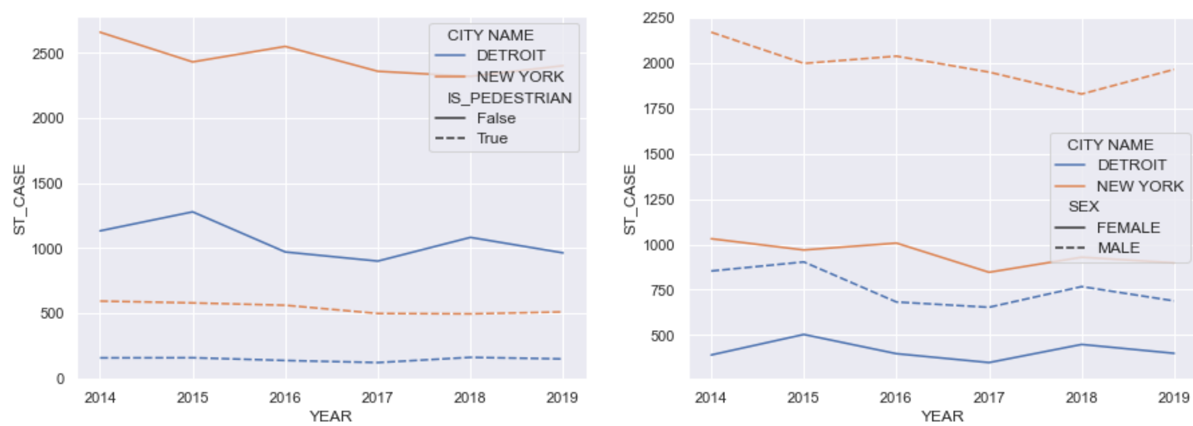




In [120]:

```
# Let's first look at whether there is a significant difference in the a
ge of people involved in fatal pedestrian crashes
# and how that may change over time
fig, axes = plt.subplots(1, 2, figsize=(15, 5))

# let's create a new column to indicate whether person involved in accid
ent was a pedestrian or not
df_p['IS_PEDESTRIAN'] = df_p.PER_TYP.map(lambda x: True if x in [4,5,6,7
,8,10,19] else (False if x in [1,2,3,9] else None))
crash_ped = df_p.groupby(['CITY NAME', 'YEAR', 'IS_PEDESTRIAN']).ST_CASE
.count().reset_index()
sns.lineplot(data=crash_ped, x='YEAR', y='ST_CASE', style='IS_PEDESTRIA
N', hue='CITY NAME', ax=axes[0])


# we will map the SEX column to more readable format
sex_map = {1:'MALE', 2:'FEMALE', 8:None, 9:None, 'MALE':'MALE', 'FEMALE'
:'FEMALE'} # there are a low enough count of unknown/not reported cases
 that we will drop them for the viz
df_p['SEX'] = df_p.SEX.map(sex_map, na_action='ignore')
crash_sex = df_p.groupby(['CITY NAME', 'YEAR', 'SEX']).ST_CASE.count().r
eset_index()
sns.lineplot(data=crash_sex, x='YEAR', y='ST_CASE', style='SEX', hue='CI
TY NAME', ax=axes[1])
```

Out[120]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd3ab464290>



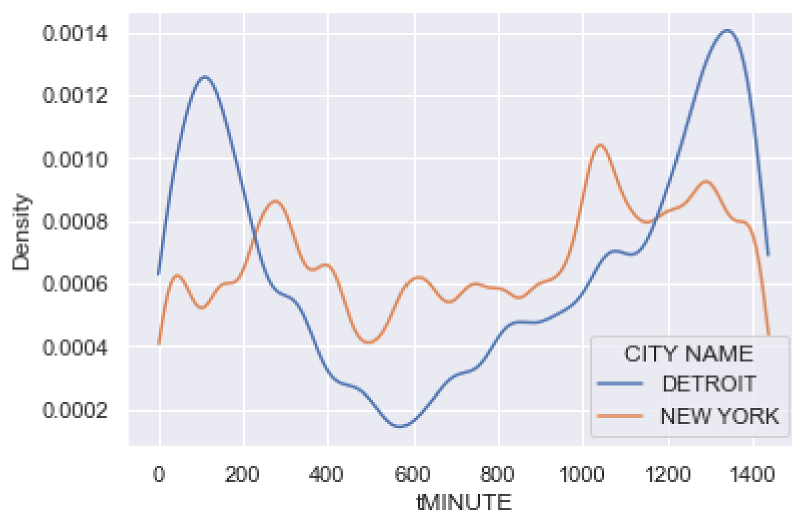
In [ ]:

In [9]:

```
# let's get the population lvl information

# we can group things by Sex, Age, number of fatalities, drug use(?)
```

In [121]:
```
# let's get the total elapsed minutes in the day for each row, so we can plot it over the course of time
df_a['tMINUTE'] = df_a.apply(lambda row: None if not (0<=row.HOUR<=23 and 0<=row.MINUTE<59) else row.HOUR*60 + row.MINUTE, axis=1)

sns.kdeplot(data=df_a, x='tMINUTE', hue='CITY NAME', cut=0, common_norm=False, bw_adjust=.3)

# we notice that there is a pretty obvious peak late at night for DETROIT between ~10pm and ~3am
## (not that there is a dip n the KDE at the ends, this is just b/c of the way the KDE is fit
## -- ideally we would use a KDE method that can handle cyclic data... a future improvement for vizualization purposes )
# Whereas for NYC the cases fluctuate but there is no dominant time of day
# There is a upward shift starting after ~5-6pm for NYC, which might relate to when people leaving work


# a future analysis might be to investigate if the increase in incidents in detroit late at night might be related to
# the type of drivers who would be driving during this time, namely commercial drivers. We can look at the cross-section of drivers
# by license type and see if this uptick in DETROIT associated with higher
```

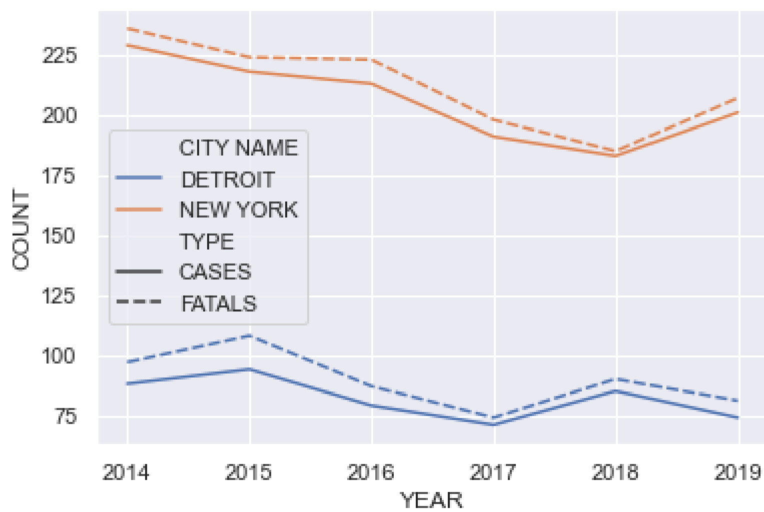
Out[121]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fd3ab471410>`



In [120]:

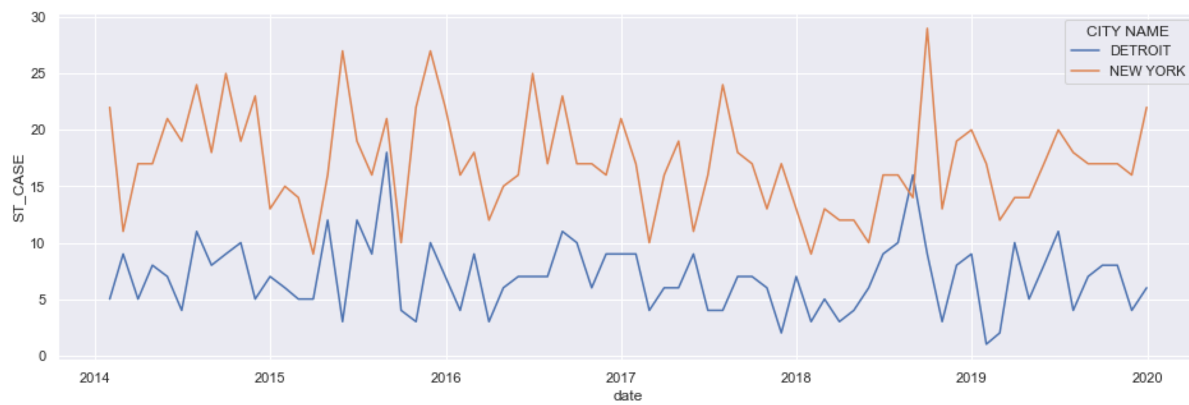
```
# We can look at the overall cases and deaths yr-over-yr for each city
# to see if there is any significant difference in trend
crash_counts = df_a.groupby(['CITY NAME', 'YEAR']).ST_CASE.count().reset
_index().assign(TYPE='CASES').rename(columns={'ST_CASE':'COUNT'})
fatal_counts = df_a.groupby(['CITY NAME', 'YEAR']).FATALS.sum().reset_in
dex().assign(TYPE='FATALS').rename(columns={'FATALS':'COUNT'})
all_counts = pd.concat([crash_counts, fatal_counts], axis=0)

sns.lineplot(data=all_counts, x="YEAR", y="COUNT", hue="CITY NAME", styl
e='TYPE')

# Both cities seem to have an overall downward trend (thought NYC has an
uptick in 2019)
# It would be interesting to see how much of this could be driven by dec
lining numbers of licensed drivers in each city
# To explore this we could join data from NHTSA showing the number of li
censed drivers per city per year for 2014-2019.
```

Out[120]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb66bbb11d0>



In [77]:

```
# we can look at more granular time series trends
df_a['date'] = df_a.apply(lambda row: datetime(year=row.YEAR, month=row.
MONTH, day=row.DAY), axis=1)

day_crash_counts = df_a.sort_values('date').groupby(['CITY NAME', 'date'
]).ST_CASE.count().reset_index()
week_crash_counts = day_crash_counts.groupby(['CITY NAME', pd.Grouper(ke
y='date', freq='W-SUN')]).ST_CASE.sum().reset_index()
month_crash_counts = day_crash_counts.groupby(['CITY NAME', pd.Grouper(k
ey='date', freq='M')]).ST_CASE.sum().reset_index()

fig, axes = plt.subplots(figsize=(16,5))
sns.lineplot(data=month_crash_counts, x="date", y="ST_CASE", hue="CITY N
AME")
plt.show()


# as a future step, we can look at seasonality (using something like ARI
MA modeling) to determine if there is any interesting
# differences in seasonal peaks between the two cities. For example, one
hypothesis is that more people visit NYC during
# the winter holiday, in which case there may be an increase in the numb
er of pedestrian fatal crashes due to the higher volume
# of pedestrians on the street and travelling around the city.
```



In [ ]: