Stats and Public Health Part 1: Cleaning and EDA

Daniel Mortensen

3-Aug-2021

Imports

Library imports

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats
```

Importing the mosquito tracking data for the West Nile Virus.

In [2]:
```
mosquito_df = pd.read_csv('./mosquito_data.csv')
display(mosquito_df.head())
```

	Year	Week	Address Block	Block	Trap	Trap type	Date	Mosquito number	Mosquito ID	WNV Present	Species
0	2019	39	100XX W OHARE AIRPORT	100	T910	GRAVID	2019-09-26 00:09:00	2	Res	negative	CULEX RESTUANS
1	2019	39	52XX S KOLMAR AVE	52	T114	GRAVID	2019-09-26 00:09:00	1	Res	negative	CULEX RESTUANS
2	2019	39	58XX N WESTERN AVE	58	T028	GRAVID	2019-09-26 00:09:00	2	Res	negative	CULEX RESTUANS
3	2019	39	39XX N SPRINGFIELD AVE	39	T228	GRAVID	2019-09-26 00:09:00	1	Res	negative	CULEX RESTUANS
4	2019	39	131XX S BRANDON AVE	131	T209	GRAVID	2019-09-26 00:09:00	9	Res	negative	CULEX RESTUANS

Instructions

West Nile Virus (WNV) is a viral illness largely spread by mosquitoes. The disease is transmitted to a person when an infected mosquito bites them.

The city of Chicago, Illinois has been keeping track of mosquito populations and WNV prevalence using a series of traps that they place around the city. They are then able to study the captured specimens and monitor the state of WNV spread in the city.

You are given mosquito tracking data from 2008 to 2019.

In this deliverable, you will perform basic EDA and data wrangling to get familiar with the dataset from the city of Chicago.

Part 1 - Basic Data Wrangling

1. What is the shape of the dataframe?

In [3]:
```
print(mosquito_df.shape)
```

```
(18495, 13)
```

The table has 13 columns, each containing a unique category of information, and 18,495 rows

2. Convert the 'Date' column to have a datetime format.

Converting the "Date" column to a datetime format using the to_datetime function.

In [4]:
```
mosquito_df["Date"] = pd.to_datetime(mosquito_df["Date"])
display(mosquito_df.sample(3))
```

	Year	Week	Address Block	Block	Trap	Trap type	Date	Mosquito number	Mosquito ID	WNV Present	Species
2516	2017	32	79XX S CHICAGO	79	T083	GRAVID	2017-08-10 00:08:00	3	Res	negative	CULEX RESTUANS
3514	2016	33	5XX S CENTRAL AVE	5	T031	GRAVID	2016-08-18 00:08:00	9	Res	negative	CULEX RESTUANS
63	2019	38	52XX W 63RD ST	52	T065	GRAVID	2019-09-19 00:09:00	1	Res	negative	CULEX RESTUANS

3. Pick two numeric and two categorical columns: What data they are storing? How are they distributed?

Numeric Columns

- Year: This column stores the year in which the trap was checked. These values range from 2007 to 2019.
- Mosquito number: this column stores how many mosquitos were found in the trap. On average, there were 11 mosquitos per trap, with a standard deviation of 13, a minimum of 1, and a maximum of 50.

Categorical Columns

- Trap type: This column describes the type of trap that was used at a given location. There are 4 unique trap types, and the GRAVID trap type is the most common, accounting for 95% of all traps.
- Species: This column describes the species of mosquito that was found in the trap. There are 4 different mosquito species that have been found in the traps. The Culex Testuans is th most common species, comprising nearly two thirds of all mosquitos trapped.

In [5]:

```python
# These lines of code are queries used to explore the data distribtutions.

print("Info about \"Year\" column")
print(mosquito_df["Year"].describe())
print("\n")

print("Info about \"Mosquito number\" column")
print(mosquito_df["Mosquito number"].describe())
print("\n")

print("Info about \"Trap type\" column")
print(mosquito_df["Trap type"].describe())
print("\n")

print(17741/18495*100)
print("\n")

print("Info about \"Species\" column")
print(mosquito_df["Species"].describe())
print("\n")

print(11866/18495*100)
```

```
Info about "Year" column
count    18495.000000
mean      2012.905812
std          3.725857
min       2007.000000
25%       2010.000000
50%       2013.000000
75%       2016.000000
max       2019.000000
Name: Year, dtype: float64


Info about "Mosquito number" column
count    18495.000000
mean        10.879913
std         13.475066
min          1.000000
25%          2.000000
50%          5.000000
75%         14.000000
max         50.000000
Name: Mosquito number, dtype: float64


Info about "Trap type" column
count      18495
unique         4
```

```
top       GRAVID
freq       17741
Name: Trap type, dtype: object


95.92322249256556


Info about "Species" column
count              18495
unique                 4
top       CULEX RESTUANS
freq               11866
Name: Species, dtype: object


64.15788050824547
```

4. Are there any columns that contain duplicate information? If so, remove the redundant columns.

In [6]:
```python
count_duplicates = mosquito_df.duplicated(keep='first').sum()
percent_duplicates = round(mosquito_df.duplicated(keep='first').sum() / mosquito_df.sha
print(f"There are {count_duplicates} duplicated rows, corresponding to {percent_duplica
```

```
There are 658 duplicated rows, corresponding to 3.56 % of the total rows in the mosquito
dataframe.
```

In [7]:
```python
mosquito_df = mosquito_df.drop_duplicates()
count_duplicates = mosquito_df.duplicated(keep='first').sum()
print(f"There are now {count_duplicates} duplicated rows.")
```

```
There are now 0 duplicated rows.
```

5. Are there any null values in the dataframe? If so, deal with them appropriately.

I first need to count to see if there are any null vallues and which columns they are in.

In [8]:
```python
mosquito_df.isna().sum(axis=0)
```

Out[8]:
```
Year                   0
Week                   0
Address Block          0
Block                  0
Trap                   0
Trap type              0
Date                   0
Mosquito number        0
Mosquito ID            0
WNV Present            0
Species                0
Lat                 2266
Lon                 2266
dtype: int64
```

There are only missing values for the latitude and longitude. It may be possible to replace these values if they are listed for the same stations on different days (rows).

In [9]:
```python
# I will first determine how big of a percentage this is.
percent_missing = mosquito_df["Lat"].isna().sum(axis=0) / mosquito_df.shape[0] * 100
```

```
print(percent_missing)

# I will then make a dataframe containing only the Trap, Lat, and Lon
station_locations = mosquito_df[["Trap", "Lat", "Lon"]].drop_duplicates().sort_values("

# I will then find the average location for each trap
station_locations = station_locations.groupby("Trap").mean()
display(station_locations)

# I can now figure out how many unique stations have missing longitudes and latitudes.
missing_lat = station_locations['Lat'].isna().sum()
missing_lon = station_locations['Lon'].isna().sum()
print(f'There are {missing_lon} stations with missing longitudes and latitudes. That is
```

```
12.703930033077313
```

	Lat	Lon
Trap		
220A	41.987054	-87.728398
T001	41.954282	-87.733843
T002	41.956304	-87.797512
T002A	41.965414	-87.782119
T002B	41.955269	-87.797048
...	...	...
T920	NaN	NaN
T921	NaN	NaN
T923	NaN	NaN
T924	NaN	NaN
T925	NaN	NaN

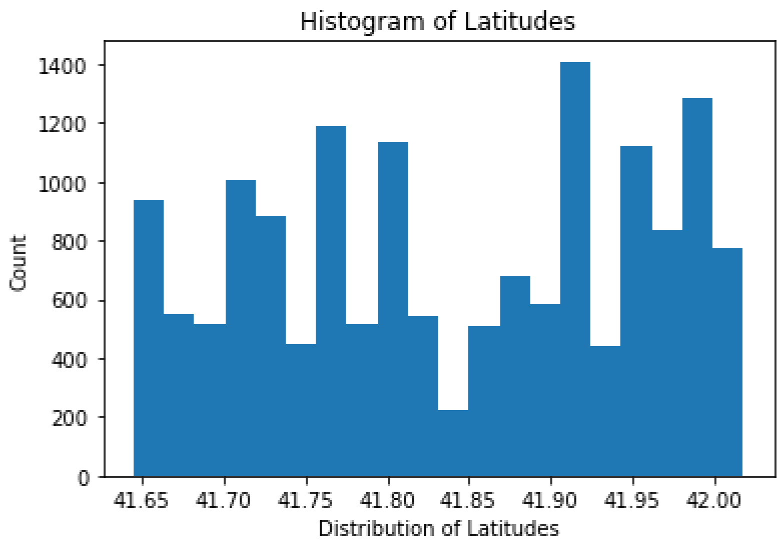
190 rows × 2 columns

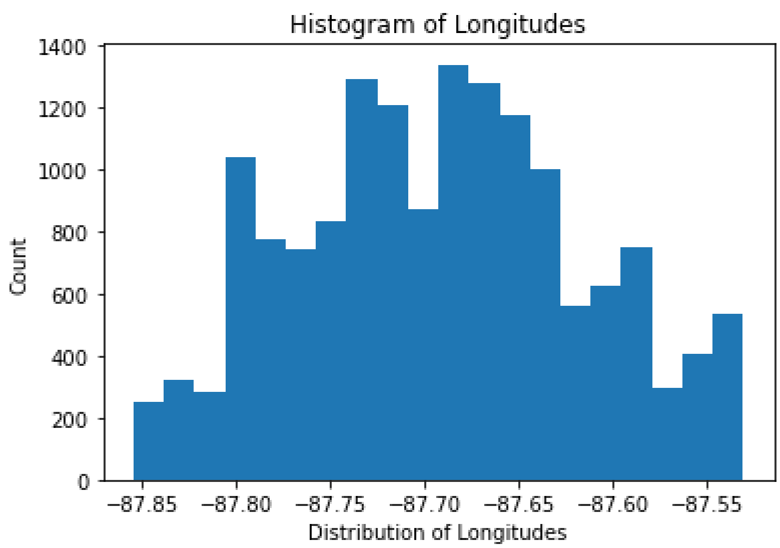
```
There are 31 stations with missing longitudes and latitudes. That is 16.3 % of the stati
ons.
```

In [10]:

```
plt.figure()
plt.hist(mosquito_df["Lat"], bins=20)
plt.title("Histogram of Latitudes")
plt.ylabel("Count")
plt.xlabel("Distribution of Latitudes")
plt.show()

plt.figure()
plt.hist(mosquito_df["Lon"], bins=20)
plt.title("Histogram of Longitudes")
plt.ylabel("Count")
plt.xlabel("Distribution of Longitudes")
plt.show()
```





Nearly 1 in 6 stations is missing its location information. Dropping this information will have sever negative consequences for my ability to draw conclusions from the data. Therefore, I will replace these values with the average longitude and latitude values from the other rows.

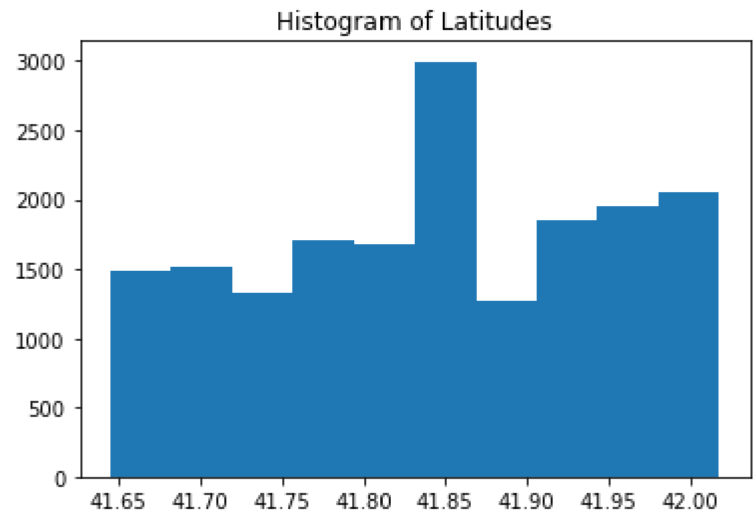
In [11]:

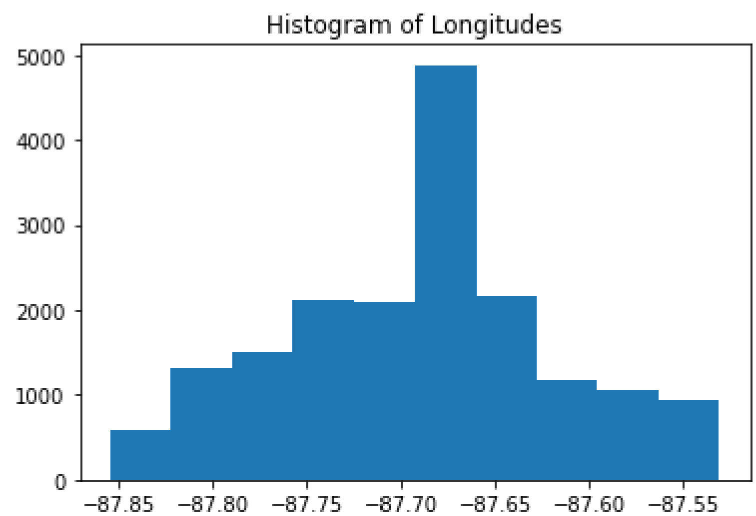
```
mosquito_df["Lat"] = mosquito_df["Lat"].fillna(mosquito_df["Lat"].mean())
mosquito_df["Lon"] = mosquito_df["Lon"].fillna(mosquito_df["Lon"].mean())
```

In [12]:

```
plt.figure()
plt.hist(mosquito_df["Lat"])
plt.title("Histogram of Latitudes")
plt.show()

plt.figure()
plt.hist(mosquito_df["Lon"])
plt.title("Histogram of Longitudes")
plt.show()
```





In [13]:

```
mosquito_df.isna().sum(axis=0)
```

Out[13]:

```
Year                0
Week                0
Address Block       0
Block               0
Trap                0
Trap type           0
Date                0
Mosquito number     0
Mosquito ID         0
WNV Present         0
Species             0
Lat                 0
Lon                 0
dtype: int64
```

All missing data has now been filled in.

Part 2 - Basic EDA

1. Using an appropriate visual, or visuals, explore the relationship between mosquito number and date.

I will make a subplot for the data from each year, broken down by month. In case new years are addded, I will make the number of plots in the subplot dynamic rather than static. For ease of comparison, I will also scale all of the y-axes to the same value.

In [14]:

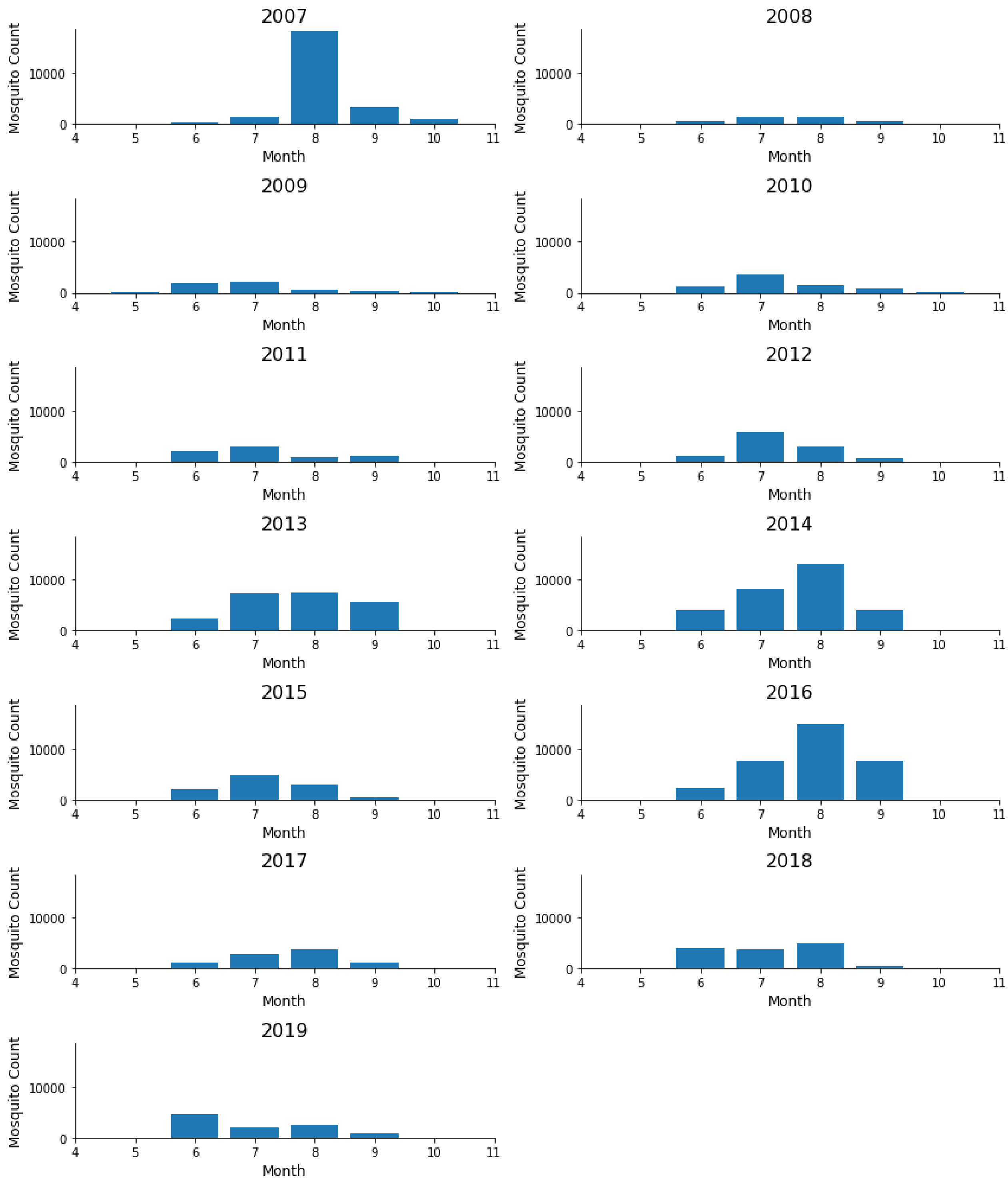
```
earliest_year = mosquito_df["Date"].dt.year.min()
last_year =  mosquito_df["Date"].dt.year.max()
year_count = last_year - earliest_year + 1

rows = int(year_count / 2 + year_count % 2)

plt.subplots(year_count, figsize=(12,14))
for i in range(0, year_count):
    new_df = mosquito_df[["Date", "Mosquito number"]].where(mosquito_df["Year"] == (ear
    summed_data = new_df.groupby(new_df["Date"].dt.month).sum()

    plt.subplot(rows, 2 , i + 1)
    plt.bar(summed_data.index.values, summed_data["Mosquito number"])
    plt.xlabel("Month", size=12)
    plt.ylabel("Mosquito Count", size=12)
    plt.xlim([4,11])
    plt.ylim([0,18500])
    plt.title(f"{earliest_year + i}", size = 16)

sns.despine()
plt.tight_layout()
plt.show()
```



Part 3 - Advanced EDA

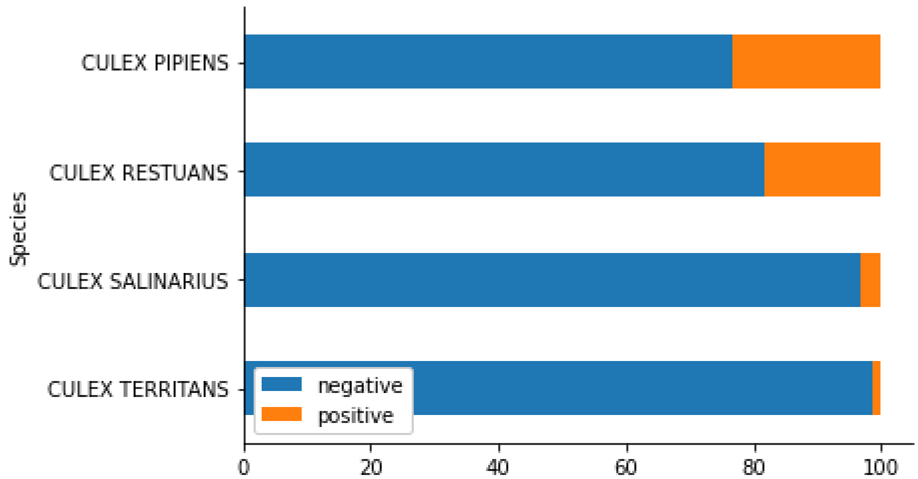
1. Using an appropriate visual, explore the relationship between mosquito species and WNV prevalence.

In [15]:

```
total_count = mosquito_df.groupby(["Species", "WNV Present"])["WNV Present"].count()
pct_infected = (total_count / mosquito_df.groupby(["Species"])["WNV Present"].count() *

pct_infected.sort_values("positive").plot(kind="barh", stacked=True)
plt.legend(framealpha = 1, loc=3)
```

```
sns.despine()
plt.show()
```



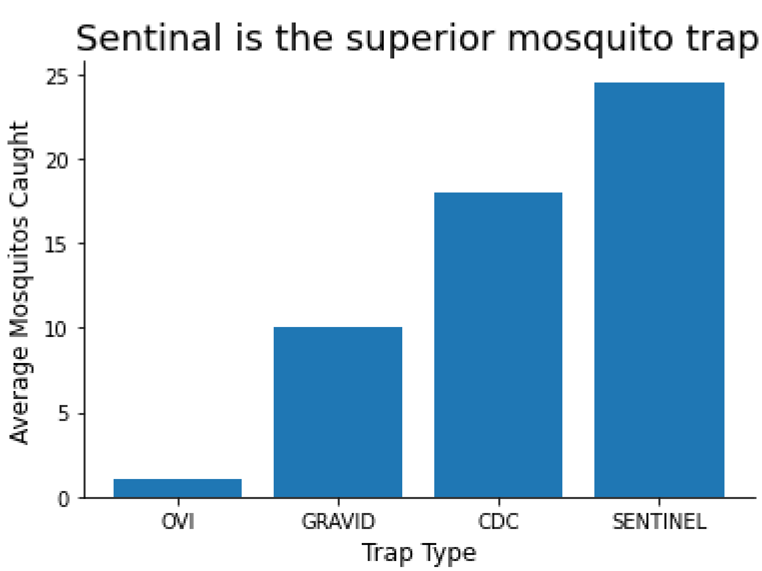
2. Using an appropriate visual, explore the relationship between the number of mosquitos caught and trap type.

Note: This visual should be a different type of visualization than the previous one

In [16]:

```
number_by_trap_df = mosquito_df[["Trap type", "Mosquito number"]].groupby("Trap type").
number_by_trap_df = number_by_trap_df.sort_values("Mosquito number", ascending=True)

plt.figure()
plt.bar(number_by_trap_df.index.values, number_by_trap_df["Mosquito number"])
sns.despine()
plt.xlabel("Trap Type", size=12)
plt.ylabel("Average Mosquitos Caught", size=12)
plt.title("Sentinal is the superior mosquito trap", size=18)
plt.show()
```



3. Using an appropriate visual, come up with an additional insight of your choice.

Note: This visual should be a different type of visualization than the previous two

In [17]:

```
species_year_df = mosquito_df[["Year", "Species", "Mosquito number"]].groupby(["Year",
species_year_df = species_year_df.unstack()

plt.figure()
plt.scatter(species_year_df.index.values, species_year_df["Mosquito number"]["CULEX PIP
plt.plot(species_year_df.index.values, species_year_df["Mosquito number"]["CULEX PIPIEN

plt.scatter(species_year_df.index.values, species_year_df["Mosquito number"]["CULEX RES
plt.plot(species_year_df.index.values, species_year_df["Mosquito number"]["CULEX RESTUA

plt.scatter(species_year_df.index.values, species_year_df["Mosquito number"]["CULEX SAL
plt.plot(species_year_df.index.values, species_year_df["Mosquito number"]["CULEX SALINA

plt.scatter(species_year_df.index.values, species_year_df["Mosquito number"]["CULEX TER
plt.plot(species_year_df.index.values, species_year_df["Mosquito number"]["CULEX TERRIT

plt.legend()
plt.xlabel("Year", size=12)
plt.ylabel("Total number of Mosquitoes Caught")
plt.title("\"RESTUANS\" the current dominant species")

sns.despine()
plt.show()
```