Stats and Public Health Part 2: Data Analysis

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
import statsmodels.api as sm

from scipy import stats
from statsmodels.stats.proportion import proportions_ztest
```

Importing the mosquito tracking data for the West Nile Virus.

In [2]:

```
mosquito_original = pd.read_csv('./mosquito_data_part_2.csv')
mosquito_df = mosquito_original.copy()
```

In [3]:

```
mosquito_df.shape
```

Out[3]:

```
(18495, 12)
```

In [4]:

```
mosquito_df.isna().sum()
```

Out[4]:

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

Data Management Plan

- Year - Numeric --> **no change**
- Week - Numeric --> could change to cyclic variable, but we don't need weekly granularity for this report --> **drop column**

- Address Block - Categorical --> data is repeated in `Lon` and `Lat` columns --> **drop column**
- Trap - Categorical --> Essentially synonymous with `Address Block` and `Lon` and `Lat` columns --> **drop column**
- Trap type - Categorical --> **Convert to dummy variable**
- Date - Numeric --> Already represented by `Year`, `Month` and `Week` columns --> **drop column**
- Mosquito number - Numeric --> **no change**
- WNV Present - Categorical --> **convert to binary**
- Species - Categorical --> **convert to dummy variables**
- Lat - Numeric --> **no change**
- Lon - Numeric --> **no change**
- Month - Numeric --> **change to cyclic variable**

In [5]:

```
mosquito_df.drop(columns=["Week", "Address Block", "Trap", "Date"], inplace=True)
```

Determining column data types and splitting them into a numeric data frame and a categorical data frame for processing.

In [6]:

```
print(mosquito_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18495 entries, 0 to 18494
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Year             18495 non-null  int64  
 1   Trap type        18495 non-null  object 
 2   Mosquito number  18495 non-null  int64  
 3   WNV Present      18495 non-null  object 
 4   Species          18495 non-null  object 
 5   Lat              18495 non-null  float64
 6   Lon              18495 non-null  float64
 7   Month            18495 non-null  int64  
dtypes: float64(2), int64(3), object(3)
memory usage: 1.1+ MB
None
```

In [7]:

```
df_numeric = mosquito_df.select_dtypes(["int64", "float64", "uint8", "int32"])
df_categorical = mosquito_df.select_dtypes("object")
```

Instructions

Now that you are familiar with the data, we will move on to a set of analyses on the relationship between the different variables and the mosquito number, as well as the probability of finding West Nile Virus (WNV) at any particular time and location.

Part 1 - Basic Analysis

1. Convert the `WNV Present` column into a binary column, and create dummy variables from the `Trap type` column.

Determining possible values for `WNV Present`

In [8]:
```
df_categorical["WNV Present"].value_counts()
```

Out[8]:
```
negative    14501
positive     3994
Name: WNV Present, dtype: int64
```

Possible values include "negative" and "positive". Replacing "negative" with "0" and "positive" with "1".

In [9]:
```
WNV_neg = (df_categorical['WNV Present'] == "negative")
WNV_zero = (df_categorical['WNV Present'] == 0)

df_numeric['WNV Present'] = np.where(WNV_neg | WNV_zero, 0, 1)
mosquito_df['WNV Present'] = np.where(WNV_neg | WNV_zero, 0, 1)
df_categorical.drop(columns="WNV Present", inplace=True)
display(df_categorical.head(5))
```

```
<ipython-input-9-9f628e3d596f>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df_numeric['WNV Present'] = np.where(WNV_neg | WNV_zero, 0, 1)
C:\Users\Daniel\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

	Trap type	Species
0	GRAVID	CULEX RESTUANS
1	GRAVID	CULEX RESTUANS
2	GRAVID	CULEX RESTUANS
3	GRAVID	CULEX RESTUANS
4	GRAVID	CULEX RESTUANS

Creating dummy variables for trap type. There is not a clear best trap type to drop, so I will simply drop the first one, which is `CDC`

In [10]:
```
trap_type_dummies = pd.get_dummies(df_categorical[["Trap type"]], drop_first=True)

df_numeric = pd.concat([df_numeric, trap_type_dummies], axis = 1)
df_numeric.head(3)
df_categorical.drop(columns="Trap type", inplace=True)
```

```
C:\Users\Daniel\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

Determining possible Species and their counts

In [11]:

```
df_categorical.value_counts(normalize=True)*100
```

Out[11]:

```
Species
CULEX RESTUANS      64.157881
CULEX PIPIENS       29.662071
CULEX TERRITANS      4.958097
CULEX SALINARIUS     1.221952
dtype: float64
```

Changing `Species` to a dummy variable. Species CULEX SALINARIUS is the least populus, representing only about 1% of all mosquitos caught, so I will drop the dummy variable for this species.

In [12]:

```
species_dummies = pd.get_dummies(df_categorical[["Species"]]).drop(columns="Species_CUL
df_numeric = pd.concat([df_numeric, species_dummies], axis = 1)
df_numeric.head(3)
df_categorical.drop(columns="Species", inplace=True)
```

For the month, we will use cyclic encoding.

In [13]:

```
base_angle = 2*np.pi/12
print(base_angle)

cos = np.cos(df_numeric["Month"]*base_angle)
sin = np.sin(df_numeric["Month"]*base_angle)

df_numeric["Month_cos"] = cos
df_numeric["Month_sin"] = sin
```

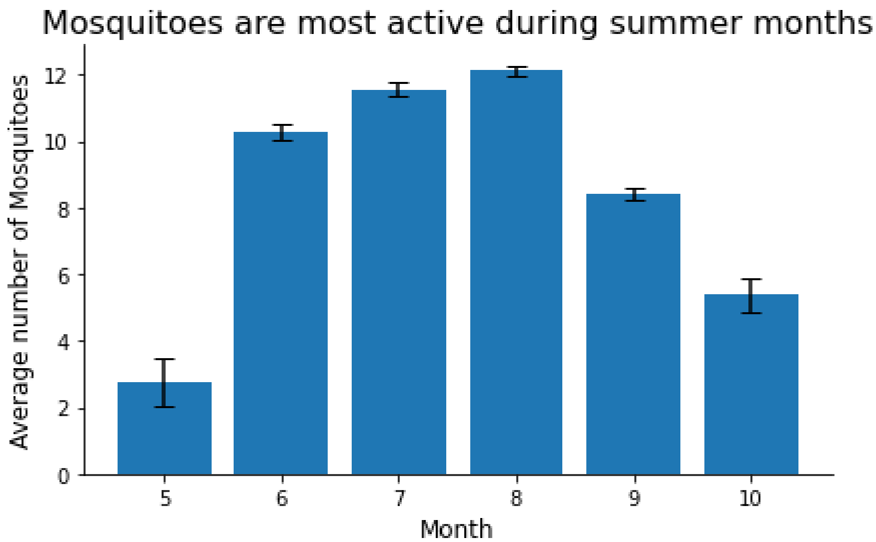
```
0.5235987755982988
```

2. What is the average number of mosquitoes for each month? What trends do you notice?

In [14]:

```
monthy_averages = df_numeric.groupby("Month").mean()
monthly_sem = df_numeric.groupby("Month").sem()

plt.figure()
plt.bar(monthy_averages.index.values, monthy_averages["Mosquito number"], yerr=monthly_
plt.xlabel("Month", size=12)
plt.ylabel("Average number of Mosquitoes", size=12)
plt.title("Mosquitoes are most active during summer months", size=16)
sns.despine()
plt.tight_layout()
plt.show()
```



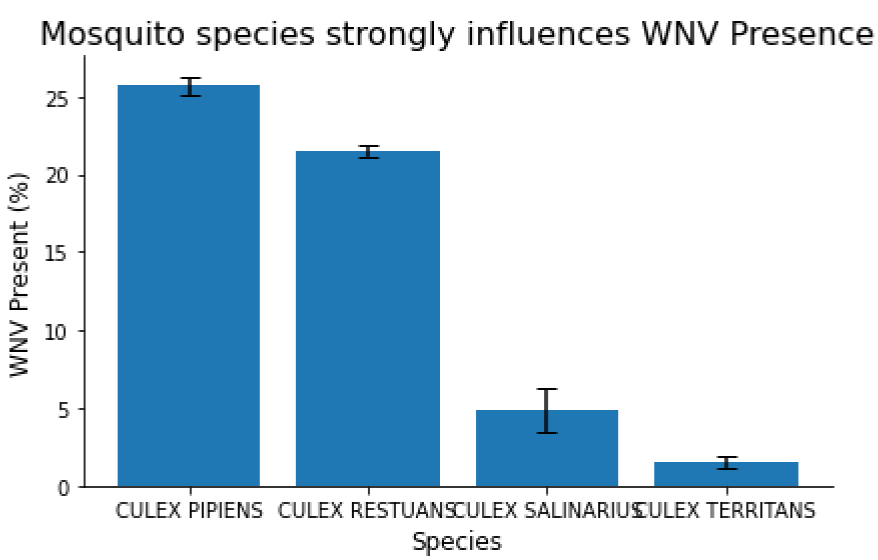
Part 2 - Statistical Analysis

1. Is there a statistically significant difference between the different mosquito species when looking at the occurrence of West Nile Virus?

In [15]:

```
species_averages = mosquito_df.groupby("Species")["WNV Present"].aggregate(["mean", "st

plt.figure()
plt.bar(species_averages.index.values, species_averages["mean"]*100, yerr=species_avera
plt.xlabel("Species", size=12)
plt.ylabel("WNV Present (%)", size=12)
plt.title("Mosquito species strongly influences WNV Presence", size=16)
sns.despine()
plt.tight_layout()
plt.show()
```



I will first test to see if there is any significance between the populations using an ANOVA test.

In [16]:

```
anova_data = {}
mosquitoes = mosquito_df["Species"].unique()
```

```
for species in mosquitoes:
    anova_data[species] = mosquito_df.loc[mosquito_df["Species"] == species, "WNV Prese

stats.f_oneway(anova_data["CULEX PIPIENS"],
              anova_data["CULEX RESTUANS"],
              anova_data["CULEX SALINARIUS"],
              anova_data["CULEX TERRITANS"])
```

Out[16]:
```
F_onewayResult(statistic=105.45270503888439, pvalue=1.082615440825039e-67)
```

Results from ANOVA indicate that there is at least one species which is significantly different from the others. I will now run a proportions z test between each species pair individually.

In [17]:
```
mosquitoes = mosquito_df["Species"].unique()

sums = []
for key in anova_data:
    sums.append(anova_data[key].sum())

counts = []
for key in anova_data:
    counts.append(anova_data[key].count())

for n in range(4):
    for m in range(n+1, 4):
        compare_sums = []
        compare_sums.append(sums[n])
        compare_sums.append(sums[m])

        compare_counts = []
        compare_counts.append(counts[n])
        compare_counts.append(counts[m])

        print("p value (" + mosquitoes[n] + " vs " + mosquitoes[m] + "):")
        print(proportions_ztest(compare_sums, compare_counts)[1])
        print("\n")
```

```
p value (CULEX RESTUANS vs CULEX TERRITANS):
4.149911157085309e-48


p value (CULEX RESTUANS vs CULEX SALINARIUS):
1.2461111009600203e-09


p value (CULEX RESTUANS vs CULEX PIPIENS):
1.0019767964463832e-09


p value (CULEX TERRITANS vs CULEX SALINARIUS):
0.0021029008936009263


p value (CULEX TERRITANS vs CULEX PIPIENS):
8.231567115719057e-60


p value (CULEX SALINARIUS vs CULEX PIPIENS):
1.164688112052138e-12
```

The proportions z tests allow me to conclude that there is a significant difference between all species.

2. Which columns are positively correlated with the number of mosquitoes caught? Which columns are negatively correlated? Are these correlations statistically significant?

I will determine the linear correlation between `Mosquito number` and all other variables using the `corr()` function.

In [18]:
```
correlation = df_numeric.corr()
correlation = correlation["Mosquito number"]
correlation.pop("Mosquito number")
correlation = correlation.sort_values(ascending=False)
correlation
```

Out[18]:
```
WNV Present              0.408034
Year                     0.129326
Trap type_SENTINEL       0.108575
Lat                      0.096820
Species_CULEX RESTUANS   0.070999
Species_CULEX PIPIENS    0.014730
Month_sin                0.005443
Trap type_OVI           -0.005392
Month                   -0.040426
Month_cos               -0.064980
Trap type_GRAVID        -0.138275
Species_CULEX TERRITANS -0.150962
Lon                     -0.151421
Name: Mosquito number, dtype: float64
```

The following columns have a positive correlation with `Mosquito number`:

- WNV Present
- Year
- Trap type_SENTINEL
- Lat
- Species_CULEX RESTUANS
- Species_CULEX PIPIENS
- Month_sin

The following columns have a negative correlation with `Mosquito number`:

- Trap type_OVI
- Month
- Month_cos
- Trap type_GRAVID
- Species_CULEX TERRITANS
- Lon

The strongest correlation is with `WNV Present`, which only has a correlation value of about 0.41, indicating that none of these values has a strong linear correlation with `Mosquito number`. It should be noted that ion this experiment, we are only looking at a linear correlation. Trying to correlate these data using other function types, such as logarithems, may result in stronger correlation values.

Part 3 - Advanced Statistical Analysis

1. Run a linear regression to determine how the independent variables affect the number of mosquitoes caught. Explain your model construction process. Analyze the model and the results, and discuss the model's limitations. This may end up being an iterative process.

Note:

- You will likely see a low R^2 value, that is to be expected.
- This dataset does not respond well to performing VIF analysis, so this is not required.
- `WNV Present` **must not** be one of your independent variables.

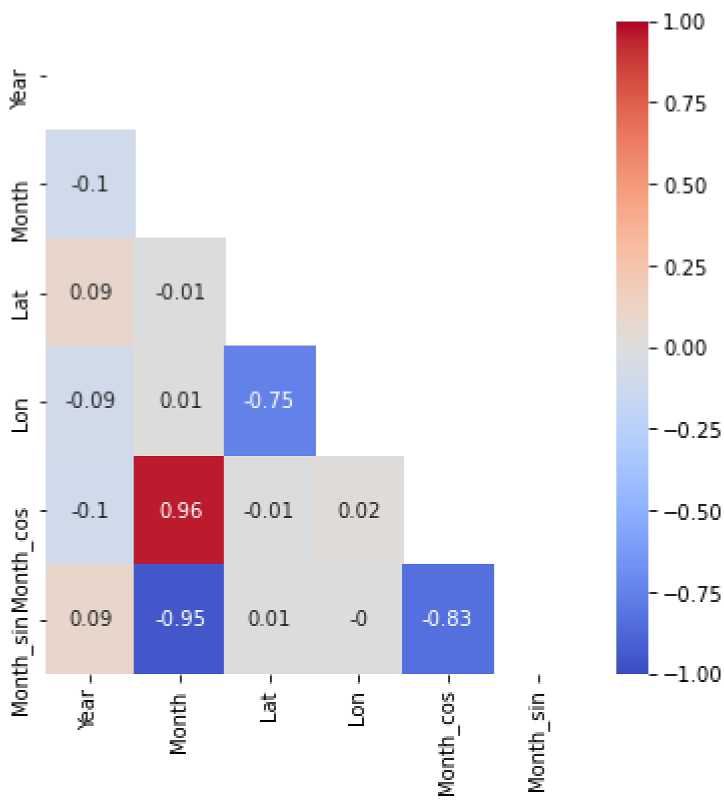
In [20]:

```
x_vals = df_numeric[["Year", "Month", "Lat", "Lon", "Month_cos", "Month_sin"]]
y_vals = df_numeric["Mosquito number"]
```

In [21]:

```
df_corr = x_vals.corr()

plt.figure(figsize=(6,6))
sns.heatmap(df_corr.round(2),  vmin=-1, vmax=1, cmap="coolwarm", annot=True, mask=np.tr
plt.show()
```



In [22]:

```
x_vals_const = sm.add_constant(x_vals)

mosquito_lr = sm.OLS(y_vals, x_vals_const)
mosquito_lr = mosquito_lr.fit()
display(mosquito_lr.summary())
```

OLS Regression Results

Dep. Variable:	Mosquito number	**R-squared:**	0.047
Model:	OLS	**Adj. R-squared:**	0.047
Method:	Least Squares	**F-statistic:**	152.5
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	1.10e-189
Time:	17:11:21	**Log-Likelihood:**	-73899.
No. Observations:	18495	**AIC:**	1.478e+05
Df Residuals:	18488	**BIC:**	1.479e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-2715.6779	111.802	-24.290	0.000	-2934.820	-2496.536
Year	0.4063	0.026	15.473	0.000	0.355	0.458
Month	-0.2387	1.540	-0.155	0.877	-3.258	2.780
Lat	-4.9906	1.267	-3.940	0.000	-7.474	-2.508
Lon	-24.0731	1.533	-15.706	0.000	-27.078	-21.069
Month_cos	-6.4370	2.518	-2.556	0.011	-11.372	-1.502
Month_sin	-6.5608	2.182	-3.007	0.003	-10.837	-2.284

Omnibus:	5024.299	**Durbin-Watson:**	1.538
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	10667.030
Skew:	1.622	**Prob(JB):**	0.00
Kurtosis:	4.820	**Cond. No.**	2.33e+06

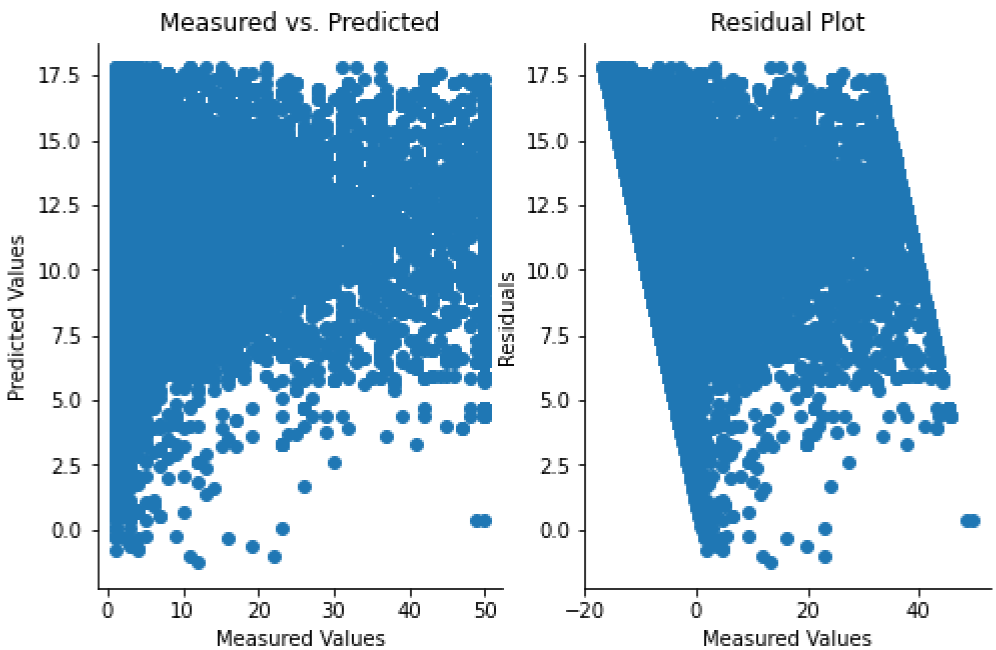
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.33e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [23]:

```
# soft predictions, the probability of getting the deposit
y_pred = mosquito_lr.predict(x_vals_const)

plt.subplots(2, figsize=(8,5))
```

```
plt.subplot(1,2,1)
plt.scatter(y_vals, y_pred)
plt.xlabel("Measured Values")
plt.ylabel("Predicted Values")
plt.title("Measured vs. Predicted")
plt.subplot(1,2,2)
plt.scatter(mosquito_lr.resid, mosquito_lr.fittedvalues)
plt.xlabel("Measured Values")
plt.ylabel("Residuals")
plt.title("Residual Plot")
sns.despine()
plt.show()
```



In [24]:

```
x_vals = df_numeric[["Year", "Lon", "Month_cos"]]
y_vals = df_numeric["Mosquito number"]

x_vals_const = sm.add_constant(x_vals)
mosquito_lr = sm.OLS(y_vals, x_vals_const)
mosquito_lr = mosquito_lr.fit()
display(mosquito_lr.summary())

plt.subplots(2, figsize=(8,5))
plt.subplot(1,2,1)
plt.scatter(y_vals, y_pred)
plt.xlabel("Measured Values")
plt.ylabel("Predicted Values")
plt.title("Measured vs. Predicted")
plt.subplot(1,2,2)
plt.scatter(mosquito_lr.resid, mosquito_lr.fittedvalues)
plt.xlabel("Measured Values")
plt.ylabel("Residuals")
plt.title("Residual Plot")
sns.despine()
plt.show()
```

OLS Regression Results

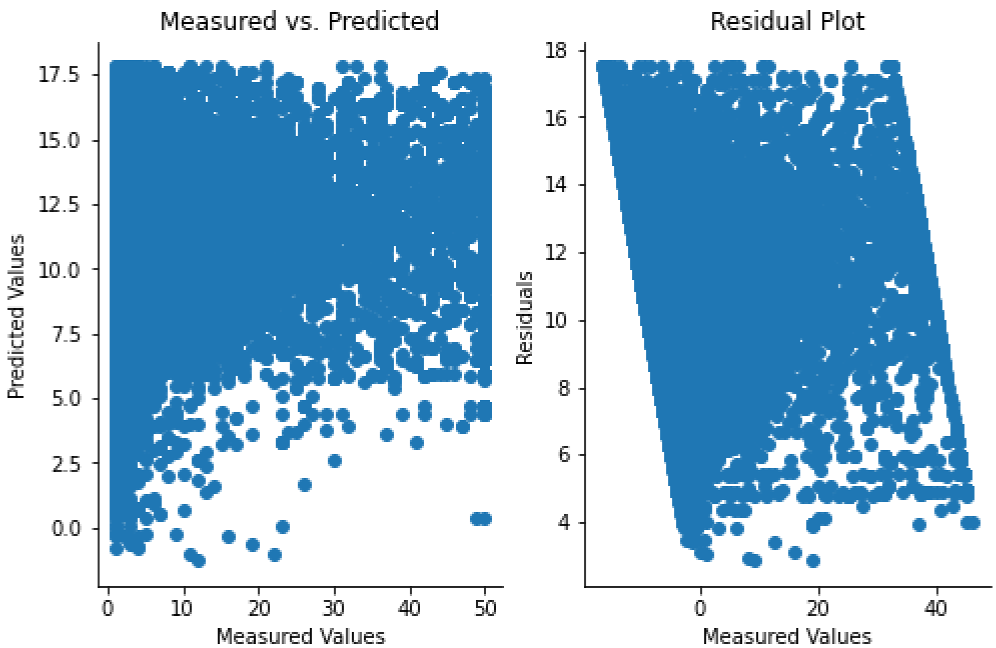
Dep. Variable:	Mosquito number	**R-squared:**	0.039

Model:	OLS	**Adj. R-squared:**	0.039
Method:	Least Squares	**F-statistic:**	249.5
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	9.31e-159
Time:	17:11:21	**Log-Likelihood:**	-73978.
No. Observations:	18495	**AIC:**	1.480e+05
Df Residuals:	18491	**BIC:**	1.480e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-2533.2062	99.826	-25.376	0.000	-2728.873	-2337.539
Year	0.4010	0.026	15.235	0.000	0.349	0.453
Lon	-19.7895	1.023	-19.348	0.000	-21.794	-17.785
Month_cos	-1.9387	0.272	-7.126	0.000	-2.472	-1.405

Omnibus:	5098.704	**Durbin-Watson:**	1.523
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	10941.515
Skew:	1.640	**Prob(JB):**	0.00
Kurtosis:	4.854	**Cond. No.**	2.07e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.07e+06. This might indicate that there are
strong multicollinearity or other numerical problems.



2. Run a logistic regression to determine how the independent variables affect West Nile Virus presence. Explain your model construction process. Analyze the model and the results, and discuss the model's limitations. This may end up being an iterative process.

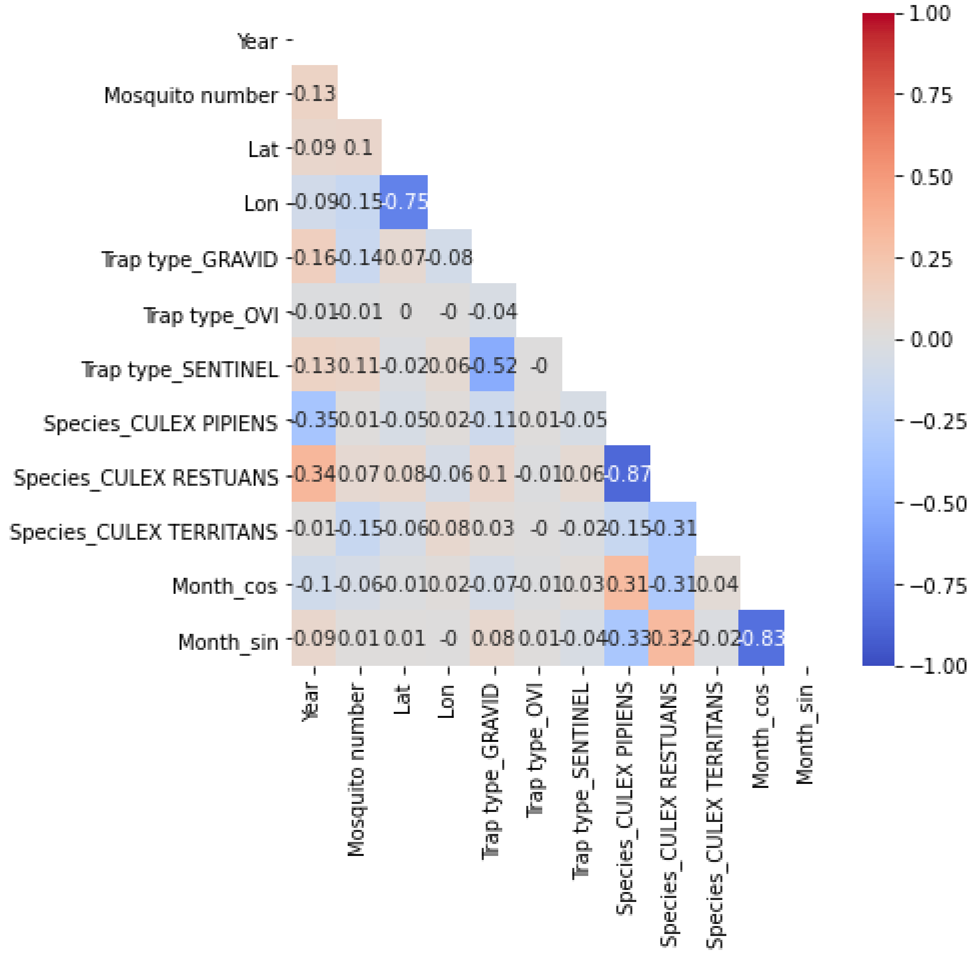
Note: `Mosquito number` should be one of your independent variables.

In [25]:

```
x_vals_logis = df_numeric.drop(columns=["WNV Present", "Month"])
x_vals_logis_const = sm.add_constant(x_vals_logis)
y_vals_logis = df_numeric["WNV Present"]

df_corr_logis = x_vals_logis.corr()

plt.figure(figsize=(6,6))
sns.heatmap(df_corr_logis.round(2),  vmin=-1, vmax=1, cmap="coolwarm", annot=True, mask
plt.show()
```



In [26]:

```
WNV_logistic_r = sm.Logit(y_vals_logis, x_vals_logis_const)
WNV_logistic_r = WNV_logistic_r.fit()
display(WNV_logistic_r.summary())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.370386
         Iterations: 35
C:\Users\Daniel\anaconda3\lib\site-packages\statsmodels\base\model.py:566: ConvergenceWa
rning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Logit Regression Results

Dep. Variable:	WNV Present	**No. Observations:**	18495
Model:	Logit	**Df Residuals:**	18482
Method:	MLE	**Df Model:**	12
Date:	Mon, 09 Aug 2021	**Pseudo R-squ.:**	0.2901
Time:	17:11:22	**Log-Likelihood:**	-6850.3
converged:	False	**LL-Null:**	-9649.5
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-574.0507	26.244	-21.874	0.000	-625.488	-522.614

Year	0.1237	0.007	17.707	0.000	0.110	0.137
Mosquito number	0.0680	0.002	43.020	0.000	0.065	0.071
Lat	-0.7723	0.297	-2.604	0.009	-1.353	-0.191
Lon	-3.9552	0.349	-11.334	0.000	-4.639	-3.271
Trap type_GRAVID	0.2650	0.138	1.926	0.054	-0.005	0.535
Trap type_OVI	-11.1220	6682.078	-0.002	0.999	-1.31e+04	1.31e+04
Trap type_SENTINEL	-0.2750	0.223	-1.231	0.218	-0.713	0.163
Species_CULEX PIPIENS	1.0406	0.327	3.181	0.001	0.400	1.682
Species_CULEX RESTUANS	0.8890	0.327	2.722	0.006	0.249	1.529
Species_CULEX TERRITANS	-1.3407	0.424	-3.163	0.002	-2.171	-0.510
Month_cos	-3.3889	0.149	-22.700	0.000	-3.681	-3.096
Month_sin	-7.0518	0.218	-32.282	0.000	-7.480	-6.624

In [27]:

```
# soft predictions, the probability of getting the deposit
y_proba = WNV_logistic_r.predict(x_vals_logis_const)

y_pred = np.where(y_proba >= 0.5, 1, 0)

num_correct = (y_pred == y_vals_logis).sum()

accuracy = num_correct / y_pred.shape[0] * 100

print(f"The model accuracy is {round(accuracy,2)}%")
```

```
The model accuracy is 83.04%
```

Model	Accuracy
Model v1	83.04%

All trap types have p values above 5%. I will try dropping all of them.

In [28]:

```
x_vals_logis = df_numeric.drop(columns=["WNV Present", "Month", "Trap type_GRAVID", "Tr
x_vals_logis_const = sm.add_constant(x_vals_logis)
y_vals_logis = df_numeric["WNV Present"]

WNV_logistic_r = sm.Logit(y_vals_logis, x_vals_logis_const)
WNV_logistic_r = WNV_logistic_r.fit()

y_proba = WNV_logistic_r.predict(x_vals_logis_const)
y_pred = np.where(y_proba >= 0.5, 1, 0)
num_correct = (y_pred == y_vals_logis).sum()
accuracy = num_correct / y_pred.shape[0] * 100
print(f"The model accuracy is {round(accuracy,2)}%")
```

```
Optimization terminated successfully.
         Current function value: 0.370737
         Iterations 9
The model accuracy is 82.92%
```

Model	Accuracy

Model	Accuracy
Model v1	83.04%
Model v2	82.92%

Slight loss in accuracy when trap type is removed. Trap type will stay in the model. I will also try dropping longitude and latitude, seperately.

In [29]:

```
x_vals_logis = df_numeric.drop(columns=["WNV Present", "Month", "Lon"])
x_vals_logis_const = sm.add_constant(x_vals_logis)
y_vals_logis = df_numeric["WNV Present"]

WNV_logistic_r = sm.Logit(y_vals_logis, x_vals_logis_const)
WNV_logistic_r = WNV_logistic_r.fit()

y_proba = WNV_logistic_r.predict(x_vals_logis_const)
y_pred = np.where(y_proba >= 0.5, 1, 0)
num_correct = (y_pred == y_vals_logis).sum()
accuracy = num_correct / y_pred.shape[0] * 100
print(f"The model accuracy w/o Lon is {round(accuracy,2)}%")

x_vals_logis = df_numeric.drop(columns=["WNV Present", "Month", "Lat"])
x_vals_logis_const = sm.add_constant(x_vals_logis)
y_vals_logis = df_numeric["WNV Present"]

WNV_logistic_r = sm.Logit(y_vals_logis, x_vals_logis_const)
WNV_logistic_r = WNV_logistic_r.fit()

y_proba = WNV_logistic_r.predict(x_vals_logis_const)
y_pred = np.where(y_proba >= 0.5, 1, 0)
num_correct = (y_pred == y_vals_logis).sum()
accuracy = num_correct / y_pred.shape[0] * 100
print(f"The model accuracy w/o Lat is {round(accuracy,2)}%")
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.373923
         Iterations: 35
The model accuracy w/o Lon is 82.68%
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.370570
         Iterations: 35
The model accuracy w/o Lat is 82.95%
```

```
C:\Users\Daniel\anaconda3\lib\site-packages\statsmodels\base\model.py:566: ConvergenceWa
rning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
C:\Users\Daniel\anaconda3\lib\site-packages\statsmodels\base\model.py:566: ConvergenceWa
rning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

Model	Accuracy
Model v1	83.04%
Model v2	82.92%
Model v3a	82.68%
Model v3b	82.95%

Model v1 is the most accurate.