Avis Analysis

Chris Sucic
Sponsored by: Dr. Amber Wagner and Dr. Scot Duncan

5/11/20

Abstract

The population of birds in the United States has decreased by 30% since 1970. With such a drastic decline in population, it is clear that more research is needed to determine what could be causing such a disaster. To help contribute to the body of research, this project looked at whether drought levels had an impact on warbler sightings in the southeastern United States. Warblers were chosen as the family of birds to research because they are particularly sensitive to changes in their environment, and the Southeast was chosen as the area of study because two migration pathways pass through the area. It was found that, generally, an increase in drought level in an area led to a decrease in warbler sightings in the same area. But, it was also found that for 17 of 36 species of warblers examined an extreme drought was needed to see a statistically significant change in the number of warblers sighted. On the opposite end, for 8 species abnormally dry conditions were enough to change the number of warblers sighted. This suggests a difference of impact on different species, and further research should be conducted to determine if there are commonalities among these differently impacted species.

Table of Contents

Introduction

According to a recent study, the bird population in the United States has decreased by roughly thirty percent since 1970. This has caused the birding community to try and find new ways to preserve and grow our remaining bird populations. One way of contributing to this effort is to further bird research by investigating what could be causing the bird population decline. While researching for this project, I did not find any studies on the impact of drought on bird populations. So, I decided to look into whether drought conditions had on impact on the number of birds sighted, which is a good indicator of bird population. Warblers were chosen as the family of birds to research because they are particularly sensitive to changes in their environment, and the Southeast was chosen as the area of study because two migration pathways pass through the area.

When starting the project, the hypothesis was that we would see a negative relationship between drought levels and warbler sightings, meaning that an increase in drought levels in an area would mean a decrease in warbler sightings. This was found to be the case generally. But, an unexpected result was encountered that suggests different species of warblers are effected by drought levels differently. This is expanded upon later in the report.

Technical Documentation

The two data sources for this project were eBird and the United States Drought Monitor. eBird is a website used by bird watchers to log their bird sightings and is run by the Cornell Ornithology department. The United States Drought Monitor is the agency responsible for assessing drought levels in the United States.

This section is broken up into 3 subsections: eBird Data Processing, Drought Data Processing, and Linear Regression Algorithm.

eBird Data Processing

After downloading bird sighting data from eBird, the first step is to filter the dataset. This is done using an R library called Auk, which is specifically built for the eBird dataset. The filters applied for this dataset were the following: only pulling data from southeastern states, only pulling warbler species, and limiting the date range. The date range for our pulled data was from 2002-2019. After this process, there were 18 files with all warbler sightings from each year in southeastern states.

Each of these 18 files were further filtered and aggregated using Python and the Pandas library. One of the biggest actions taken during this process was finding a rate called observations/minute. This variable is the number of birds seen per minute spent birding for each reported sighting. If the number of bird observations were not translated into a rate, then the number of birds observed would be skewed because eBird's user base has grown over time, leading to higher bird counts overall. At the end of the process, we have the average monthly observations/minute for each species by state and year. This process is shown in the code in Appendix A.

Drought Data Processing

Drought data was acquired for each southeastern state for the years 2002-2019. The average percentage of the state under each drought level, D0 (abnormally dry) - D4 (extreme), for each month was calculated. From here, the two datasets were brought together based on matching

states and months. or each warbler species there is the average number of sightings per minute of observation for each month and the average area of the state under each drought level and total drought level. The code for this process can be found in Appendix B. An example of the combined bird and drought dataset can be found in Appendix C.

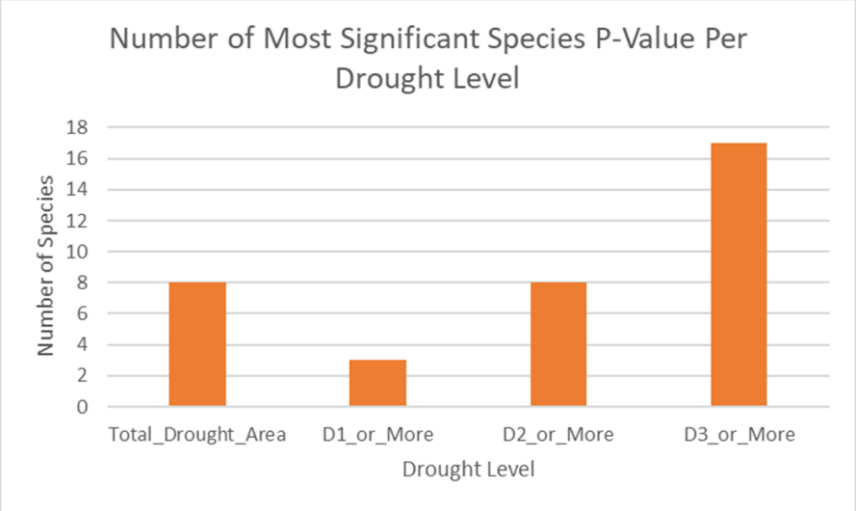
Linear Regression Algorithm

Using the combined bird data and drought data dataset, a linear regression was run on the columns observations/minute and a drought level for every combination of species, states, months, and drought level. This totaled 3000 linear regressions and statistical values associated with them such as R-Value, R^2 Value, and P-Value. The libraries used for this analysis included Pandas, Numpy, ScikitLearn, and matplotlib. The code for the linear regression can be found in Appendix D.

After running this linear regression in Python, I looked at the results in Excel and a trend that was not obvious by looking at the data appeared. This is expanded upon in the next section.

Evaluation

Analyzing the linear regressions, it is found that 70% of them have a negative R-Value, meaning that there is a negative relationship between drought levels and warbler sightings. This is what was hypothesized, so in that sense this project was successful. There was an interesting result though beyond the regression, and the chart below shows it.



The P-Value is a statistical measurement of the significance of a linear regression result. The lower a P-Value, the more significant the result is. For this project, if the P-Value was below 0.1, it was considered significant. On the chart, the most significant P-Value for each species of warbler is categorized by drought level. It is shown that for 17 out of 36 species of warbler, their most significant P-Value occurs at a drought level of D3 or higher. That means for 17 species, a high drought level is required to see a change in the number of warblers observed. On the opposite side of the graph, for 8 species of warblers a drought level of D0 or more, or

total drought area, is required to change the number of observations, meaning that they are highly sensitive to even abnormally dry conditions. This species stratification of drought response was not expected and is a strong result that can be investigated further.

Future Work

A next step for this project would be to research if a category of warbler is particularly sensitive or insensitive to drought, such as southeastern coastal breeders.
Another future line of research could be to compare fall and spring migration months. This would give a better picture of migratory species response to drought, rather than migratory species and breeding species response.
Were this process to be abstracted a bit more, it should be possible to build this process into a program for ornithologists to use to quickly clean birding data. For example, an ornithologist could download the eBird dataset and filter, clean, and prepare the dataset quickly with this program.

Appendix A: eBird Data Processing Code

```
path = r"C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior
Research/eBird_Warbler_Files_CSV"
all_files = glob.glob(path + "/*.csv")

#This is needed for us to append our looped data
li = []

#This loops over each file in our file folder
for filename in all_files:
   #Reads in CSV to a dataframe
   df = pd.read_csv(filename, low_memory = False, index_col = [0])
   #Drops columns that are not useful
   df_drop = df.drop(columns = ['LAST EDITED DATE','TAXONOMIC
ORDER','CATEGORY','SUBSPECIES COMMON NAME',
                     'SUBSPECIES SCIENTIFIC NAME','BREEDING BIRD ATLAS CODE','BREEDING
BIRD ATLAS CATEGORY',
                     'AGE/SEX','COUNTRY','COUNTRY CODE','IBA CODE','BCR CODE','USFWS
CODE',
                     'ATLAS BLOCK','LOCALITY','LOCALITY ID','LOCALITY
TYPE','LATITUDE','LONGITUDE',
                     'OBSERVER ID','SAMPLING EVENT IDENTIFIER','PROJECT CODE','HAS
MEDIA','REASON',
                     'TRIP COMMENTS','SPECIES COMMENTS'])
   #Sets a variable that holds rows that have a value of X for the column OBSERVATION
COUNT
   presence_only = df_drop[df_drop['OBSERVATION COUNT'] == 'X'].index
   #Drops rows that have a value of X for the column OBSERVATION COUNT
   df_drop.drop(presence_only, inplace = True)
   #Sets a variable that holds rows that have a 0 value for the column APPROVED
   indices = df_drop[df_drop['APPROVED'] == 0].index
   #Drops rows that have a 0 value for the column APPROVED
   df_drop.drop(indices, inplace = True)
   #Sets a dataframe where rows that have a NaN value in the column GROUP IDENTIFIER are
dropped
   df_groupid_drop = df_drop.dropna(subset = ['GROUP IDENTIFIER'])
   #Sets a variable that contains T/F null values for the column GROUP IDENTIFIER
   filled_rows = df_drop['GROUP IDENTIFIER'].notnull()
   #Sets a variable that contains the rows of df_drop for which the value of filled_rows is True
   dropped_rows = df_drop[filled_rows].index
   #Sets a dataframe that contains the rows for which the column GROUP IDENTIFIER is null
   df_blank_groupid = df_drop.drop(dropped_rows)
   #Drops duplicate rows, considers all columns exact the first
   df_groupid_duplicate_drop = df_groupid_drop.drop_duplicates(subset = ['COMMON
NAME','SCIENTIFIC NAME','OBSERVATION COUNT','STATE CODE',
```

```
                    'OBSERVATION DATE','TIME OBSERVATIONS STARTED','PROTOCOL
CODE','DURATION MINUTES',
                    'EFFORT DISTANCE KM','EFFORT AREA HA','NUMBER
OBSERVERS','ALL SPECIES REPORTED',
                    'GROUP IDENTIFIER','APPROVED','REVIEWED'])
  #Duplicating the dataframe df_blank_groupid into df_combined
  df_combined = df_blank_groupid.copy()
  #Appends the dataframe df_groupid_duplicate_drop to df_combined
  df_combined = df_combined.append(df_groupid_duplicate_drop)
  #Resets the index of df_combined
  df_combined = df_combined.reset_index()
  #Drops an unneccesary named index
  df_combined.drop(columns = ['index'], inplace = True)
  #Making a copy of our dataframe in another variable
  df_testing = df_combined.copy()
  #Dropping rows where the duration minutes == NaN
  df_testing = df_testing.dropna(subset = ['DURATION MINUTES'])
  #Dropping rows where observations are incidental or random
  incidentals = df_testing[df_testing['PROTOCOL TYPE'] == 'Incidental'].index
  randoms = df_testing[df_testing['PROTOCOL TYPE'] == 'Random'].index
  df_testing.drop(incidentals, inplace = True)
  df_testing.drop(randoms, inplace = True)
  #Finding the rate of observations/minute by dividing the observation count by the duration
minutes
  df_testing['Observations/Minute'] = df_testing['OBSERVATION COUNT'].astype(float) /
df_testing['DURATION MINUTES']
  #Creating a new dataframe with only needed columns
  data = [df_testing['COMMON NAME'], df_testing['STATE'], df_testing['OBSERVATION
DATE'], df_testing['Observations/Minute'], df_testing['OBSERVATION COUNT']]
  headers = ['COMMON NAME', 'STATE', 'OBSERVATION DATE', 'Observations/Minute',
'OBSERVATION COUNT']
  df_5col = pd.concat(data, axis = 1, keys = headers)
  #Converting the observation count to float for later calculations
  df_5col['OBSERVATION COUNT'] = df_5col['OBSERVATION COUNT'].astype(float)
  #Calculates the mean of observations/minute for matching values of common name, state,
and observation date.
  #This gives us the average daily observations/minute for each species by state
  three_cols = ['COMMON NAME', 'STATE', 'OBSERVATION DATE']
  df_5col = df_5col.groupby(three_cols).mean().reset_index()
  #Sends our observation date column to datetime for reference later
  df_5col['OBSERVATION DATE'] = pd.to_datetime(df_5col['OBSERVATION DATE'])
  #Creates separate columns for the day, month, and year
  df_5col['Day'] = df_5col['OBSERVATION DATE'].dt.day
  df_5col['Month'] = df_5col['OBSERVATION DATE'].dt.month
  df_5col['Year'] = df_5col['OBSERVATION DATE'].dt.year
  #Drops uneccesary columns
```

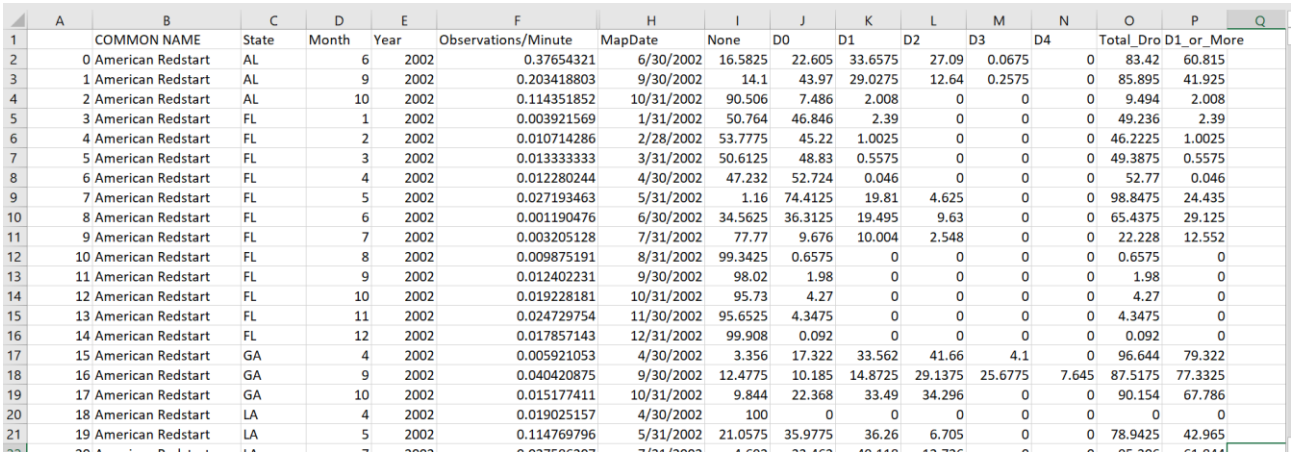
```
    df_5col.drop(columns = ['OBSERVATION DATE', 'Day'], inplace = True)
    #Calculates the mean for observations/minute for matching values of common name, state, month, and year
    #This gives us the average monthly observations/minute for each species by state and year
    four_cols = ['COMMON NAME', 'STATE', 'Month', 'Year']
    df_5col = df_5col.groupby(four_cols).mean().reset_index()
    #Appending our values to a list for later dataframe building
    li.append(df_5col)

frame = pd.concat(li, axis=0, ignore_index=True)
frame.to_csv("C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior Research/02_to_19_Warblers_Mean_original.csv")
```

Appendix B: Drought Data Processing Code

```
#-----------------------------------------------------
#           Combining Drought CSVs
#-----------------------------------------------------


path = r"C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior Research/Drought_Data"
all_files = glob.glob(path + "/*.csv")

li = []

for filename in all_files:
   #Getting the year from the name of the file
   name = Path(filename).stem
   year = int(name[10:14])
   #Reading in our drought data file
   df_drought = pd.read_csv(filename)
   #Drops StatisticFormatID column
   df_drought.drop(columns = ['StatisticFormatID'], inplace = True)
   #Creates list of columns to sum to find total area of drought in each state per week
   total_cols = ['D0', 'D1', 'D2', 'D3', 'D4']
   d1_cols = ['D1', 'D2', 'D3', 'D4']
   #Adds columns that sums total drought area of the state and total area of state above D1
drought level
   df_drought['Total_Drought_Area'] = df_drought[total_cols].sum(axis = 1)
   df_drought['D1_or_More'] = df_drought[d1_cols].sum(axis = 1)
   #Converts columns to datetime format for easier aggregation
   df_drought['MapDate'] = df_drought['MapDate'].astype(str)
   df_drought['MapDate'] = pd.to_datetime(df_drought['MapDate'])
   df_drought['ValidStart'] = pd.to_datetime(df_drought['ValidStart'])
   df_drought['ValidEnd'] = pd.to_datetime(df_drought['ValidEnd'])
   #Columns ValidStart and ValidEnd will be dropped for now, I may use them later
   df_drought.drop(columns = ['ValidStart', 'ValidEnd'], inplace = True)
   #This filters out any year not in the dataset
   df_drought = df_drought[df_drought['MapDate'].dt.year == year]
   #Averages the monthly percentage area under drought for each state
   columns = [pd.Grouper(key = 'MapDate', freq = 'M'), 'StateAbbreviation']
   df_drought = df_drought.groupby(columns).mean().reset_index()
   #Appending to our list for later use in concat
   li.append(df_drought)

#This creates a
frame = pd.concat(li, axis=0, ignore_index=True)
frame.to_csv("C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior
Research/02_to_19_Drought_Data.csv")
```

```
#-----------------------------------------------------
#        Combining Bird and Drought Data
#-----------------------------------------------------


df_5col = pd.read_csv("C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior
Research/02_to_19_Warblers_Mean_original.csv", low_memory = False, index_col = [0])
df_drought = pd.read_csv("C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior
Research/02_to_19_Drought_Data.csv", low_memory = False, index_col = [0])
us_state_abbrev = {
'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR', 'California': 'CA', 'Colorado': 'CO',
'Connecticut': 'CT', 'Delaware': 'DE', 'Florida': 'FL', 'Georgia': 'GA', 'Hawaii': 'HI', 'Idaho': 'ID',
'Illinois': 'IL', 'Indiana': 'IN', 'Iowa': 'IA', 'Kansas': 'KS', 'Kentucky': 'KY', 'Louisiana': 'LA',
'Maine': 'ME', 'Maryland': 'MD', 'Massachusetts': 'MA', 'Michigan': 'MI', 'Minnesota': 'MN',
'Mississippi': 'MS',
'Missouri': 'MO', 'Montana': 'MT', 'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH', 'New
Jersey': 'NJ',
'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC', 'North Dakota': 'ND', 'Ohio': 'OH',
'Oklahoma': 'OK',
'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island': 'RI', 'South Carolina': 'SC', 'South Dakota':
'SD',
'Tennessee': 'TN', 'Texas': 'TX', 'Utah': 'UT', 'Vermont': 'VT', 'Virginia': 'VA', 'Washington': 'WA',
'West Virginia': 'WV', 'Wisconsin': 'WI', 'Wyoming': 'WY'}
df_5col['STATE'] = df_5col['STATE'].map(us_state_abbrev)
df_5col.rename(columns = {"STATE" : 'State'}, inplace = True)
df_drought.rename(columns = {"StateAbbreviation" : "State"}, inplace = True)
df_drought['MapDate'] = df_drought['MapDate'].astype(str)
df_drought['MapDate'] = pd.to_datetime(df_drought['MapDate'])
df_drought['Month'] = df_drought['MapDate'].dt.month
df_drought['Year'] = df_drought['MapDate'].dt.year
#This merges df_5col and df_drought based on the States and Month matching
df_5col = pd.merge(df_5col, df_drought, on=['State', 'Month', 'Year'], how='left')
df_5col.to_csv("C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior
Research/02_to_19_Mean_Merge.csv")
```

Appendix C: Combined Dataset Example



	A	B	C	D	E	F	H	I	J	K	L	M	N	O	P	Q
1		COMMON NAME	State	Month	Year	Observations/Minute	MapDate	None	D0	D1	D2	D3	D4	Total_Dro	D1_or_More	
2	0	American Redstart	AL	6	2002	0.37654321	6/30/2002	16.5825	22.605	33.6575	27.09	0.0675	0	83.42	60.815	
3	1	American Redstart	AL	9	2002	0.203418803	9/30/2002	14.1	43.97	29.0275	12.64	0.2575	0	85.895	41.925	
4	2	American Redstart	AL	10	2002	0.114351852	10/31/2002	90.506	7.486	2.008	0	0	0	9.494	2.008	
5	3	American Redstart	FL	1	2002	0.003921569	1/31/2002	50.764	46.846	2.39	0	0	0	49.236	2.39	
6	4	American Redstart	FL	2	2002	0.010714286	2/28/2002	53.7775	45.22	1.0025	0	0	0	46.2225	1.0025	
7	5	American Redstart	FL	3	2002	0.013333333	3/31/2002	50.6125	48.83	0.5575	0	0	0	49.3875	0.5575	
8	6	American Redstart	FL	4	2002	0.012280244	4/30/2002	47.232	52.724	0.046	0	0	0	52.77	0.046	
9	7	American Redstart	FL	5	2002	0.027193463	5/31/2002	1.16	74.4125	19.81	4.625	0	0	98.8475	24.435	
10	8	American Redstart	FL	6	2002	0.001190476	6/30/2002	34.5625	36.3125	19.495	9.63	0	0	65.4375	29.125	
11	9	American Redstart	FL	7	2002	0.003205128	7/31/2002	77.77	9.676	10.004	2.548	0	0	22.228	12.552	
12	10	American Redstart	FL	8	2002	0.009875191	8/31/2002	99.3425	0.6575	0	0	0	0	0.6575	0	
13	11	American Redstart	FL	9	2002	0.012402231	9/30/2002	98.02	1.98	0	0	0	0	1.98	0	
14	12	American Redstart	FL	10	2002	0.019228181	10/31/2002	95.73	4.27	0	0	0	0	4.27	0	
15	13	American Redstart	FL	11	2002	0.024729754	11/30/2002	95.6525	4.3475	0	0	0	0	4.3475	0	
16	14	American Redstart	FL	12	2002	0.017857143	12/31/2002	99.908	0.092	0	0	0	0	0.092	0	
17	15	American Redstart	GA	4	2002	0.005921053	4/30/2002	3.356	17.322	33.562	41.66	4.1	0	96.644	79.322	
18	16	American Redstart	GA	9	2002	0.040420875	9/30/2002	12.4775	10.185	14.8725	29.1375	25.6775	7.645	87.5175	77.3325	
19	17	American Redstart	GA	10	2002	0.015177411	10/31/2002	9.844	22.368	33.49	34.296	0	0	90.154	67.786	
20	18	American Redstart	LA	4	2002	0.019025157	4/30/2002	100	0	0	0	0	0	0	0	
21	19	American Redstart	LA	5	2002	0.114769796	5/31/2002	21.0575	35.9775	36.26	6.705	0	0	78.9425	42.965	

Appendix D: Linear Regression Algorithm Code

```
r2_values = []
state_list = ['AL', 'GA', 'FL', 'LA', 'MS', 'NC', 'SC', 'VA']
species = ["Ovenbird","Worm-eating Warbler","Louisiana Waterthrush",
        "Northern Waterthrush","Golden-winged Warbler","Blue-winged Warbler",
        "Black-and-white Warbler","Prothonotary Warbler","Swainson's Warbler",
        "Tennessee Warbler","Orange-crowned Warbler","Nashville Warbler",
        "Connecticut Warbler","Mourning Warbler","Kentucky Warbler",
        "Common Yellowthroat","Hooded Warbler","American Redstart","Cape May Warbler",
        "Cerulean Warbler","Northern Parula","Magnolia Warbler","Bay-breasted Warbler",
        "Blackburnian Warbler","Yellow Warbler","Chestnut-sided Warbler",
        "Blackpoll Warbler","Black-throated Blue Warbler","Palm Warbler","Pine Warbler",
        "Yellow-rumped Warbler","Yellow-throated Warbler","Prairie Warbler",
        "Black-throated Green Warbler","Canada Warbler","Wilson's Warbler"]
droughts = ['Total_Drought_Area', 'D1_or_More', 'D2_or_More', 'D3_or_More']
for drought in droughts:
   for state in state_list:
      for i in range(1, 13):
         for bird in species:
            df_merge = pd.read_csv("C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior
Research/02_to_19_Mean_Merge.csv", low_memory = False, index_col = [0])
            df = df_merge[(df_merge['Month'] == i) & (df_merge['State'] == state) &
(df_merge['COMMON NAME'] == bird) & (df_merge['Year'] >= 2008)].copy()
            print(df['Year'])
            d2_cols = ['D2', 'D3', 'D4']
            d3_cols = ['D3', 'D4']
            df["D2_or_More"] = df[d2_cols].sum(axis = 1)
            df["D3_or_More"] = df[d3_cols].sum(axis = 1)
            #print(df)
            df.replace([np.inf, -np.inf], np.nan, inplace = True)
            df.dropna(inplace = True)
```

```
if len(df) < 10:
   print(state, i, bird)
   continue
#print(df)
#We don't need this section since we're not dealing with all warblers
# drops = ['COMMON NAME', 'MapDate', 'State']
# df.drop(columns = drops, inplace = True)
# columns = ['Year', 'Month']
# df = df.groupby(columns).mean()
# #print(df)
#Defining the independent and dependent variables for regression
# X = df[drought].values.reshape(-1, 1)
y = df["Observations/Minute"].values.reshape(-1, 1)
#Running the linear regression
lm = linear_model.LinearRegression()
lm.fit(X,y)
y_pred = lm.predict(X)
#Finding the P Value
X_ = sm.add_constant(X)
model = sm.OLS(y,X_)
results = model.fit()
#Finding Correlation Coefficient (R Value) using pandas
pear_corr = df[drought].corr(df["Observations/Minute"])
years = ['2010', '2011', '2012', '2013',
        '2014', '2015', '2016', '2017', '2018', '2019']
#Plotting section
plt.scatter(X, y)
plt.plot(X, y_pred, color = "red")
title = "Linear Regression: FL, 8, Blackburnian Warbler"
plt.title(title)
plt.xlabel("D3_or_More")
plt.ylabel("Observations/Minute")
annotation = 'R^2 = ', r2_score(y, y_pred)
plt.annotate(annotation, xy = (6, 0.01))
pval = 'P-Val = ', results.pvalues[1]
plt.annotate(pval, xy = (6, 0.00))
for j,txt in enumerate(years):
   plt.annotate(txt, (X[j], y[j]))
plt.show()

#Finding mean of residuals
print(results.resid.mean())
#Attaching R^2 to a list for a later dataframe
r2_values.append(state)
r2_values.append(i)
r2_values.append(bird)
```

```
            r2_values.append(r2_score(y, y_pred))
            r2_values.append(results.pvalues[1])
            r2_values.append(pear_corr)
            r2_values.append(drought)
            print(state, i, bird, r2_score(y, y_pred), results.pvalues[1], pear_corr, drought)

#print(r2_values)
df_r2 = pd.DataFrame(np.array(r2_values).reshape(-1, 7), columns = ['State', 'Month',
'Species', 'R^2 Value', 'P Value', 'R value/Correlation Coeff', 'Drought Level'] )
#print(df_r2)
df_r2.to_csv("C:/Users/ccsuc/Desktop/Spring 2020 Classes/Senior Research/State Month
Species Mean R2 Val and P Val All Droughts.csv")
```