Using Citizen Science Species Occurrence Data to Study Biodiversity and Plan for the Future

Dylan Readel

February 2020

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Justo donec enim diam vulputate. Arcu vitae elementum curabitur vitae nunc sed velit dignissim sodales. Proin fermentum leo vel orci porta non pulvinar. Nascetur ridiculus mus mauris vitae ultricies leo integer. Facilisis magna etiam tempor orci eu. Vestibulum lorem sed risus ultricies. Euismod elementum nisi quis eleifend quam. Pulvinar sapien et ligula ullamcorper malesuada proin. Aenean pharetra magna ac placerat. Tempor id eu nisl nunc mi ipsum faucibus vitae. Et netus et malesuada fames ac turpis egestas sed. Duis ut diam quam nulla porttitor massa id neque aliquam. Euismod elementum nisi quis eleifend quam adipiscing vitae. Quam viverra orci sagittis eu volutpat odio.

Contents

List of Figures

1 Introduction

- To understand the biodiversity of an area, it is important to know the species of that area and how many of each species are present.
- Understanding the biodiversity of an area is useful for preparing for the future. Urban areas like Los Angeles are novel ecosystems. The data collected by citizen scientists like the data available on iNaturalist is extremely useful for identifying what species live in urban areas.
- For some species, seasonal distributions vary greatly. Database information can be utilized to understand how species come and go from an area over the course of a year. This seasonal distribution knowledge can be useful for identifying birds and insects that are still migrating through urban areas, or those that have changed courses to avoid the concrete jungles.
- Literature regarding urban ecosystem planning, especially in the face of climate change, is crucial for understanding the importance of this project. Furthermore, literature regarding the use of historic species occurrence records highlights the significance of why identifying the species in an area is critical for future planning.

2 Methods

The following code is a function that is planned to be able to analyze any species occurrence dataset. These datasets can be retrieved from database websites like Global Biodiversity Information Facility (GBIF), Integrated Digitized Biocollections (iDigBio), iNaturalist, and Paleobiology Database (PBDB). The goal of the function is to help users understand the distribution of a desired taxonomic rank throughout the seasons in an area of interest. Furthermore, the function will supply the user with a CSV output file containing the counts of each taxon depending on the taxonomic rank specified by the user. If the user wants to know the count of every family or species in the dataset, then that can be completed with this function.

Regular expressions are utilized in an inner function of the code presented below. The regular expression, along with user input, is used to extract all the occurrences of a desired year and tell the user how many occurrences were from that year.

The function will eventually do much more with the datasets to provide the user with as much information that is necessary for researching the biodiversity of an area.

```
1
2  import pandas as pd
3  import datetime
4  import matplotlib.pyplot as plt
5  from datetime import datetime
6  from collections import defaultdict
7  import os
8  import csv
9
10 def taxonomic_dataset_analysis():
11     '''
12     This function is designed to read in a species occurrence
       dataset, analyze the data, and create useful outputs for
       understanding taxon occurrences  throughout seasons. The
       function also outputs a CSV file that contains the counts of
       each unique taxon in the dataset in respect to the taxonomic
       rank specified by the user.
13
14     -----------------------------------
15
16     inputfile: a species occurrence CSV dataset
17     date_col: the CSV column corresponding to the occurrence date
18     '''
19
20     inputfile = ''
21     inputfile = input('Enter the dataset filename: ')
22
23     # assert error for no filename specified
24     assert len(inputfile) != 0, 'No filename entered.'
25     # assert error that the filename specified is not a string
26     assert type(inputfile) == str, 'Filename is not a string: %r' %
        inputfile
27
28     print('Beginning seasonal occurrence analysis.')
29
30     # read in the csv data file of species observations
31     data = pd.read_csv(inputfile)
32
33     # input for date column and assert that the date column exists
       within dataset
34     date_col = ''
35     date_col = input('Enter the observation date column header name
       : ')
36     if date_col in data.columns:
37         print('Column found!')
38     assert date_col in data.columns, 'Date column header not found!
       '
39
40     def seasonal_occurrences(data, date_col, inputfile):
41         '''
42         This inner function uses the occurrence dataset along with
       the date column specified to  create a histogram displaying
       specified taxonomic rank occurrences by season. The use of this
        is to visualize the seasonal distribution of species or to
       understand when people upload species observations to online
       databases such as iNaturalist.
43
```

```
    -------------------------------------

    User input is required to create the plot including the x-label, the y-label, and the title.
    '''

    # extract the dates column to a list
    dates = data[date_col].to_list()

    # create an empty list for dates that are separated using datetime
    dates_separated = list()

    # iterate through dates to separate them using datetime
    for i in dates:
        dates_separated.append(datetime.strptime(i, "%Y-%m-%d"))

    # create an empty list for observations by month
    months = list()

    # iterate through separated dates list to extract just the month as an integer
    for m in dates_separated:
        months.append(m.month)

    # create an empty list for observations by season
    observations_by_season = list()

    # iterate through each observation in the months list
    # by using an if statement for each season
    # if the month is in the season's numerical range
    # the season is appended to the observation_by_season list
    for x in months:
        if x in range(1, 4):
            observations_by_season.append('Winter')
        if x in range(4, 7):
            observations_by_season.append('Spring')
        if x in range(7, 10):
            observations_by_season.append('Summer')
        if x in range (10, 13):
            observations_by_season.append('Autumn')

    # create an empty default dictionary with integers
    # for the count of each observation by season
    season_counts = defaultdict(int)

    # iterate through seasons in observations_by_season list
    # add a count for each season to the season_counts dictionary
    for season in observations_by_season:
        season_counts[season] = season_counts[season] + 1

    x_label = ''
    x_label = input('Enter the label for the x-axis (something about seasons): ')
    y_label = ''
```

```
95          y_label = input('Enter the label for the y-axis (something
    about number of occurrences): ')
96          figtitle = ''
97          figtitle = input('Enter the title for your figure: ')
98          print("Sit tight, we're making the figure...")
99
100         # plot a histogram of the season_counts dictionary
101         plt.bar(season_counts.keys(), season_counts.values())
102
103         # parameters for the figure
104         plt.xlabel(x_label)
105         plt.ylabel(y_label)
106         plt.title(figtitle)
107         plt.tight_layout()
108
109         figure_save = ''
110         figure_save = input('What do you want to call your figure
    save? ')
111         plt.savefig(figure_save)
112         print("Figure successfully saved!")
113
114         print('Figure can be found at', os.path.abspath(figure_save
    ))
115
116         print('Now creating taxon count CSV file.')
117
118     def taxon_counts(data, date_col, inputfile):
119         '''
120         This inner function is useful for ouputting a CSV file
    containing the counts of a desired taxonomic rank. For example,
     if the user specifies the family taxonomic rank, then a CSV
    file will be created that gives the counts of each family in
    the occurrence dataset.
121
122         ---------------------------------------
123
124         This function is useful to understand the diversity of an
    area. An easy way to study the diversity of an area is to know
    what species are the most common or rare.
125         '''
126
127         # input for the desired taxonomic rank column
128         # counts of each taxon placed in a dictionary
129         taxon_col = ''
130         taxon_col = input('Enter the taxon column header name: ')
131         counts_dic = (data[taxon_col].value_counts()).to_dict()
132
133         # input for the CSV save file
134         outputcsv = ''
135         outputcsv = input('What do you want to call your {} counts
    save?'.format(taxon_col))
136
137         with open(outputcsv, 'w', newline = '') as csvfile:
138             writer = csv.writer(csvfile)
139             for species, counts in counts_dic.items():
140                 writer.writerow([species, counts])
141
```

```
142         print('CSV file can be found at', os.path.abspath(outputcsv
    ))
143 
144         # input for a user prompt of whether they want to know the
    count of a specific taxon
145         answer = ''
146         answer = input('Do you want to know the count of any
    specific taxon? Answer with Y or N.')
147         answer = answer.upper()
148 
149         # answering yes brings user to another inner function
150         if answer == 'Y':
151 
152             def counts_from_year(inputfile, date_col):
153                 '''
154 
155                 This inner function utilizes a regular expression
    and user input to extract  occurrences from a user specified
    year and output the number of occurrences in total from that
    year. The regular expression searches the date column for all
    matches to the user's year.
156 
157                 '''
158 
159                 dates = []
160 
161                 # reading in the data and appending the dates to a
    list
162                 with open(inputfile, 'r') as rf:
163                     data = csv.DictReader(rf, delimiter = ',')
164                     header = data.fieldnames
165                     for dic in data:
166                         dates.append(dic[date_col])
167 
168                 # input for the desired year
169                 # generation of a regular expression based on the
    year input
170                 year = ''
171                 year = input('Enter the year you want the number of
     occurrences from: ')
172                 pre_regex = '[^-\d]*'
173                 regex = year + pre_regex
174                 charRe = re.compile(regex)
175 
176                 year_specified = []
177                 year_counts = defaultdict(int)
178 
179                 # appending each year match to a list
180                 for date in dates:
181                     year_specified.append(charRe.match(date))
182 
183                 # creating a dictionary with the key as the year
    and the value as the occurrences
184                 for obs in year_specified:
185                     year_counts[obs] += 1
186 
187                 # this is required because each year match gets
```

```
    added as a new key to the dictionary
188                 # but the None key in the dictionary has a value
    that is equal to every year that
189                 # did not match the user input
190                 # the years that did not match is substracted from
    the total length of the dataset
191                 count = len(dates) - year_counts.get(None)
192
193                 print('The year {} had {} species observations.'.
    format(year, count))
194
195             counts_from_year(inputfile, date_col)
196
197         # if user answers no to prompt the analysis is completed
198         if answer == 'N':
199
200             print('Analysis completed.')
201
202     seasonal_occurrences(data, date_col, inputfile)
203     taxon_counts(data, date_col, inputfile)
204
205 taxonomic_dataset_analysis()
```

3 Results

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Justo donec enim diam vulputate. Arcu vitae elementum curabitur vitae nunc sed velit dignissim sodales. Proin fermentum leo vel orci porta non pulvinar. Nascetur ridiculus mus mauris vitae ultricies leo integer. Facilisis magna etiam tempor orci eu. Vestibulum lorem sed risus ultricies. Euismod elementum nisi quis eleifend quam. Pulvinar sapien et ligula ullamcorper malesuada proin. Aenean pharetra magna ac placerat. Tempor id eu nisl nunc mi ipsum faucibus vitae. Et netus et malesuada fames ac turpis egestas sed. Duis ut diam quam nulla porttitor massa id neque aliquam. Euismod elementum nisi quis eleifend quam adipiscing vitae. Quam viverra orci sagittis eu volutpat odio.

4 Discussion

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Justo donec enim diam vulputate. Arcu vitae elementum curabitur vitae nunc sed velit dignissim sodales. Proin fermentum leo vel orci porta non pulvinar. Nascetur ridiculus mus mauris vitae ultricies leo integer. Facilisis magna etiam tempor orci eu. Vestibulum lorem sed risus ultricies. Euismod elementum nisi quis eleifend quam. Pulvinar sapien et ligula ullamcorper malesuada proin. Aenean pharetra magna ac placerat. Tempor id eu nisl nunc mi ipsum faucibus vitae. Et netus et malesuada fames ac turpis egestas sed. Duis ut diam quam nulla porttitor massa id neque aliquam.

Euismod elementum nisi quis eleifend quam adipiscing vitae. Quam viverra orci sagittis eu volutpat odio.

5 Figures