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.

```
import pandas as pd
import datetime
import matplotlib.pyplot as plt
from datetime import datetime
from collections import defaultdict
import os
import csv

def taxonomic_dataset_analysis():
    '''
    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.

    ----------------------------------

    inputfile: a species occurrence CSV dataset
    date_col: the CSV column corresponding to the occurrence date
    '''

    inputfile = ''
    inputfile = input('Enter the dataset filename: ')

    # assert error for no filename specified
    assert len(inputfile) != 0, 'No filename entered.'
    # assert error that the filename specified is not a string
    assert type(inputfile) == str, 'Filename is not a string: %r' %
     inputfile

    print('Beginning seasonal occurrence analysis.')

    # read in the csv data file of species observations
    data = pd.read_csv(inputfile)

    # input for date column and assert that the date column exists
    within dataset
    date_col = ''
    date_col = input('Enter the observation date column header name
    : ')
    if date_col in data.columns:
        print('Column found!')
    assert date_col in data.columns, 'Date column header not found!
    '

    def seasonal_occurrences(data, date_col, inputfile):
        '''
        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.

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

```
44
45          User input is required to create the plot including the x-
    label, the y-label, and the title.
46          '''
47
48          # extract the dates column to a list
49          dates = data[date_col].to_list()
50
51          # create an empty list for dates that are separated using
    datetime
52          dates_separated = list()
53
54          # iterate through dates to separate them using datetime
55          for i in dates:
56              dates_separated.append(datetime.strptime(i, "%Y-%m-%d")
    )
57
58          # create an empty list for observations by month
59          months = list()
60
61          # iterate through separated dates list to extract just the
    month as an integer
62          for m in dates_separated:
63              months.append(m.month)
64
65          # create an empty list for observations by season
66          observations_by_season = list()
67
68          # iterate through each observation in the months list
69          # by using an if statement for each season
70          # if the month is in the season's numerical range
71          # the season is appended to the observation_by_season list
72          for x in months:
73              if x in range(1, 4):
74                  observations_by_season.append('Winter')
75              if x in range(4, 7):
76                  observations_by_season.append('Spring')
77              if x in range(7, 10):
78                  observations_by_season.append('Summer')
79              if x in range (10, 13):
80                  observations_by_season.append('Autumn')
81
82          # create an empty default dictionary with integers
83          # for the count of each observation by season
84          season_counts = defaultdict(int)
85
86          # iterate through seasons in observations_by_season list
87          # add a count for each season to the season_counts
    dictionary
88          for season in observations_by_season:
89              season_counts[season] = season_counts[season] + 1
90
91          x_label = ''
92          x_label = input('Enter the label for the x-axis (something
    about seasons): ')
93          y_label = ''
94          y_label = input('Enter the label for the y-axis (something
```

```
about number of occurrences): ')
95          figtitle = ''
96          figtitle = input('Enter the title for your figure: ')
97          print("Sit tight, we're making the figure...")
98
99          # plot a histogram of the season_counts dictionary
100         plt.bar(season_counts.keys(), season_counts.values())
101
102         # parameters for the figure
103         plt.xlabel(x_label)
104         plt.ylabel(y_label)
105         plt.title(figtitle)
106         plt.tight_layout()
107
108         figure_save = ''
109         figure_save = input('What do you want to call your figure
    save? ')
110         plt.savefig(figure_save)
111         print("Figure successfully saved!")
112
113         print('Figure can be found at', os.path.abspath(figure_save
    ))
114
115         print('Now creating taxon count CSV file.')
116
117     def taxon_counts(data, date_col, inputfile):
118         '''
119         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.
120
121         ---------------------------------------
122
123         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.
124         '''
125
126         # input for the desired taxonomic rank column
127         # counts of each taxon placed in a dictionary
128         taxon_col = ''
129         taxon_col = input('Enter the taxon column header name: ')
130         counts_dic = (data[taxon_col].value_counts()).to_dict()
131
132         # input for the CSV save file
133         outputcsv = ''
134         outputcsv = input('What do you want to call your {} counts
    save?'.format(taxon_col))
135
136         with open(outputcsv, 'w', newline = '') as csvfile:
137             writer = csv.writer(csvfile)
138             for species, counts in counts_dic.items():
139                 writer.writerow([species, counts])
140
141         print('CSV file can be found at', os.path.abspath(outputcsv
```

```
))

    # input for a user prompt of whether they want to know the count of a specific taxon
    answer = ''
    answer = input('Do you want to know the count of any specific taxon? Answer with Y or N.')
    answer = answer.upper()

    # answering yes brings user to another inner function
    if answer == 'Y':

        def counts_from_year(inputfile, date_col):
            '''

            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.

            '''

            dates = []

            # reading in the data and appending the dates to a list
            with open(inputfile, 'r') as rf:
                data = csv.DictReader(rf, delimiter = ',')
                header = data.fieldnames
                for dic in data:
                    dates.append(dic[date_col])

            # input for the desired year
            # generation of a regular expression based on the year input
            year = ''
            year = input('Enter the year you want the number of occurrences from: ')
            pre_regex = '[^-\d]*'
            regex = year + pre_regex
            charRe = re.compile(regex)

            year_specified = []
            year_counts = defaultdict(int)

            # appending each year match to a list
            for date in dates:
                year_specified.append(charRe.match(date))

            # creating a dictionary with the key as the year and the value as the occurrences
            for obs in year_specified:
                year_counts[obs] += 1

            # this is required because each year match gets added as a new key to the dictionary
```

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

The following code is intended to produce a map of the distribution of several species at once. Figure 1 shows the distribution of the three most common bird species in the LA River Watershed region presented on iNaturalist. The base layer can be changed for any region, but for this example a shapefile of LA County was used. The function is modular meaning any number of species or other taxon can be used and plotted. A new point variable can be created for each species or taxon. The function is designed to work with any dataset that includes latitude and longitude coordinates. The species are chosen using a grep command and extracting them from the dataset. This function is useful since a website like iNaturalist does not display different species as different point objects simultaneously. The map produced by this function allows the user to visualize how different species are spatially distributed across a region. The function can be modified to extract and plot different aspects of the datasets, as well. For example, instead of extracting and plotting species or other taxon, a user may want to extract iNaturalist user identification numbers and plot the distribution of them.

```
library(rgdal)
library(ggplot2)
library(dplyr)

dist_map <- function(filename) {

setwd('~/Documents/UCLA/UCLA Winter 2020/EEB C177/')

# read in bird occurrence data with coordinates
mapdata <- read.csv(filename, stringsAsFactors = FALSE)

# set a variable for each species that is to be plotted
# any number of variables can be assigned for the map
mapdata_mallard <- mapdata[grep('Anas platyrhynchos',
                                mapdata$taxon_species_name), ]
mapdata_dove <- mapdata[grep('Zenaida macroura',
                             mapdata$taxon_species_name), ]
mapdata_sparrow <- mapdata[grep('Passer domesticus',
                                mapdata$taxon_species_name), ]

# load in an LA County shapefile and set to a variable
county <- readOGR('DRP_COUNTY_BOUNDARY/DRP_COUNTY_BOUNDARY.shp',
    layer = 'DRP_COUNTY_BOUNDARY')

# useful for knowing how the county layer is being displayed
# for this shapefile, the projection used is NAD83
proj4string(county)

# find out what class the mapdata is
class(mapdata)

# creating a spatial point for the mapdata
# checking the mapdata class again
coordinates(mapdata_mallard) <- ~longitude+latitude
class(mapdata_mallard)
coordinates(mapdata_dove) <- ~longitude+latitude
class(mapdata_dove)
```

```
coordinates(mapdata_sparrow) <- ~longitude+latitude
class(mapdata_sparrow)

# checking if there is a projection for the mapdata
# it returns NA so there is not
proj4string(mapdata_mallard)
proj4string(mapdata_dove)
proj4string(mapdata_sparrow)

# manually setting the coordinate system for mapdata
# to be the same as the county shapefile
proj4string(mapdata_mallard) <- CRS("+proj=longlat +datum=NAD83")
proj4string(mapdata_dove) <- CRS("+proj=longlat +datum=NAD83")
proj4string(mapdata_sparrow) <- CRS("+proj=longlat +datum=NAD83")

# using spTransform function to project mapdata to the
# county shapefile
mapdata_mallard <- spTransform(mapdata_mallard, CRS(proj4string(
    county)))
mapdata_dove <- spTransform(mapdata_dove, CRS(proj4string(county)))
mapdata_sparrow <- spTransform(mapdata_sparrow, CRS(proj4string(
    county)))

# check to see if the mapdata and county projections match
# returns TRUE so they match
identical(proj4string(mapdata_mallard), proj4string(county))
identical(proj4string(mapdata_dove), proj4string(county))
identical(proj4string(mapdata_sparrow), proj4string(county))

# convert mapdata back to data.frame for ggplot to work with
mapdata_mallard <- data.frame(mapdata_mallard)
mapdata_dove <- data.frame(mapdata_dove)
mapdata_sparrow <- data.frame(mapdata_sparrow)

# changing the latitude and longitude names to x and y
# since we are working with x and y values now
names(mapdata_mallard)[names(mapdata_mallard)=="longitude"]<-"x"
names(mapdata_mallard)[names(mapdata_mallard)=="latitude"]<-"y"

names(mapdata_dove)[names(mapdata_dove)=="longitude"]<-"x"
names(mapdata_dove)[names(mapdata_dove)=="latitude"]<-"y"

names(mapdata_sparrow)[names(mapdata_sparrow)=="longitude"]<-"x"
names(mapdata_sparrow)[names(mapdata_sparrow)=="latitude"]<-"y"

    # setting parameters for creating the map
map <- ggplot() +
          # plot the county shapefile based on latitude and
    longitude
          geom_polygon(data = county,
                               aes(x = long, y = lat, group =
    group),
                               fill = 'grey40',
                               color = 'grey90',
                               alpha = 1) +
          # specify no x and y labels
          # define the title of the map
```

```
90          labs(x = '',
91               y = '',
92               title = 'Distribution of Common Birds in the LA
    River Watershed') +
93          # get rid of x and y ticks and text
94          # make the title bold, adjust its position
95          # adjust the position of the legend title and make it
    bold
96          theme(axis.ticks.y = element_blank(),axis.text.y =
    element_blank(),
97                axis.ticks.x = element_blank(),axis.text.x =
    element_blank(),
98                plot.title = element_text(lineheight=.8, face="bold
    ", vjust=1, hjust = .5),
99                legend.position = 'right', legend.title = element_
    text(face = 'bold')) +
100         # define the title of the legend
101         guides(fill = guide_legend(title = 'Species')) +
102         # plot the first species points
103         # setting the color and color as the name allows it to
    appear on the legend
104         # and also allows us to change the appearance using
    global setting
105         geom_point(data = mapdata_mallard,
106                    aes(x = x, y = y, color = 'Mallard', size = '
    Mallard')) +
107         # plot the second species points
108         geom_point(data = mapdata_dove,
109                    aes(x = x, y = y, color = 'Mourning Dove',
    size = 'Mourning Dove')) +
110         # plot the third species points
111         geom_point(data = mapdata_sparrow,
112                    aes(x = x, y = y, color = 'House Sparrow',
    size = 'House Sparrow')) +
113         # set the colors for each species
114         scale_color_manual(name = 'Species',
115                            values = c('Mallard'='red', 'Mourning
    Dove'='blue', 'House Sparrow'='yellow')) +
116         # set the size of each point for each species
117         scale_size_manual(name = 'Species',
118                           values = c('Mallard'=.4, 'Mourning Dove
    '=.4, 'House Sparrow'=.4)) +
119         # set the appearance to be square to avoid distortion
120         coord_equal(ratio = 1)
121
122 map
123 }
124
125 dist_map('birds.csv')
```

The following code is useful for producing a pie chart that displays the abundances of taxon present in a dataset. The iNaturalist bird dataset from the LA River Watershed was used in this code to display the relative abundances of each bird order present in that region. This visualization is shown in Figure 2.

```
library(ggplot2)

pie_chart <- function(filename) {

# set working directory as directory
setwd('~/Documents/UCLA/UCLA Winter 2020/EEB C177/')

# read in bird occurrence data
dataset <- read.csv(filename, stringsAsFactors=FALSE)

# extract all unique bird orders to a list
bird_orders <- list(dataset %>% select(taxon_order_name) %>%
    distinct %>% arrange(taxon_order_name))

# creates a pie chart that displays the relative abundances of
    orders
pie <- ggplot(dataset,
              aes(factor(1), fill=taxon_order_name)) +
  # removing the x and y labels, defining the title
  labs(x='', y='', title='Relative Abundances of Bird Orders in the
     LA River Watershed') +
  # making the title bold and adjusting its position
  # making the legend title bold and placing it on the right
  theme(plot.title = element_text(lineheight=.8, face="bold", vjust
    =1, hjust = .5),
        legend.position = 'right', legend.title = element_text(face
     = 'bold')) +
  # defining the title of the legend
  guides(fill = guide_legend(title='Orders')) +
  # setting the width of the slices
  geom_bar(width=1) +
  # changing chart to polar coordinates to display as pie chart
    instead of bars
  coord_polar(theta='y')

pie
}

pie_chart('birds.csv')
```

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

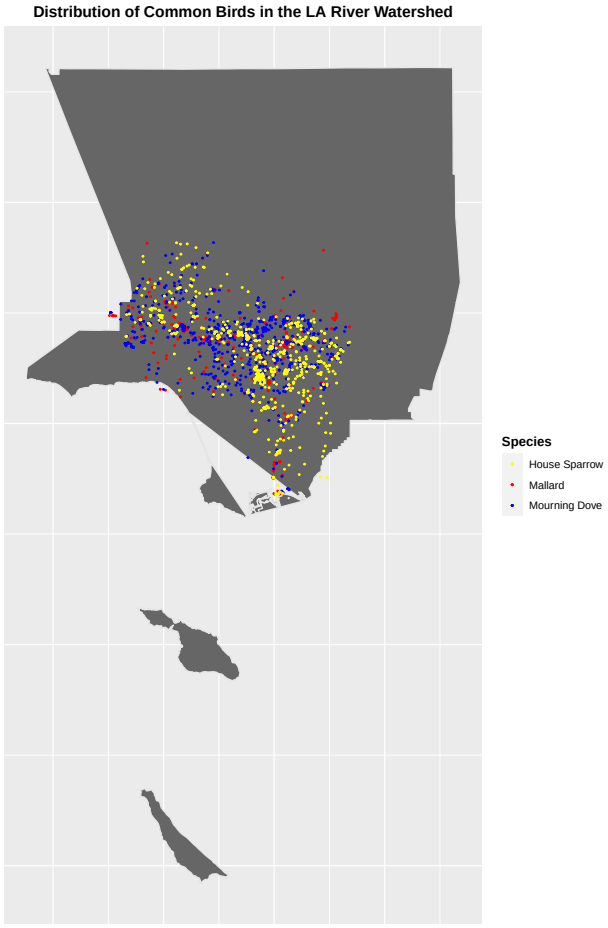

Figure 1: Distribution of common bird species within the LA River Watershed region based on iNaturalist data.

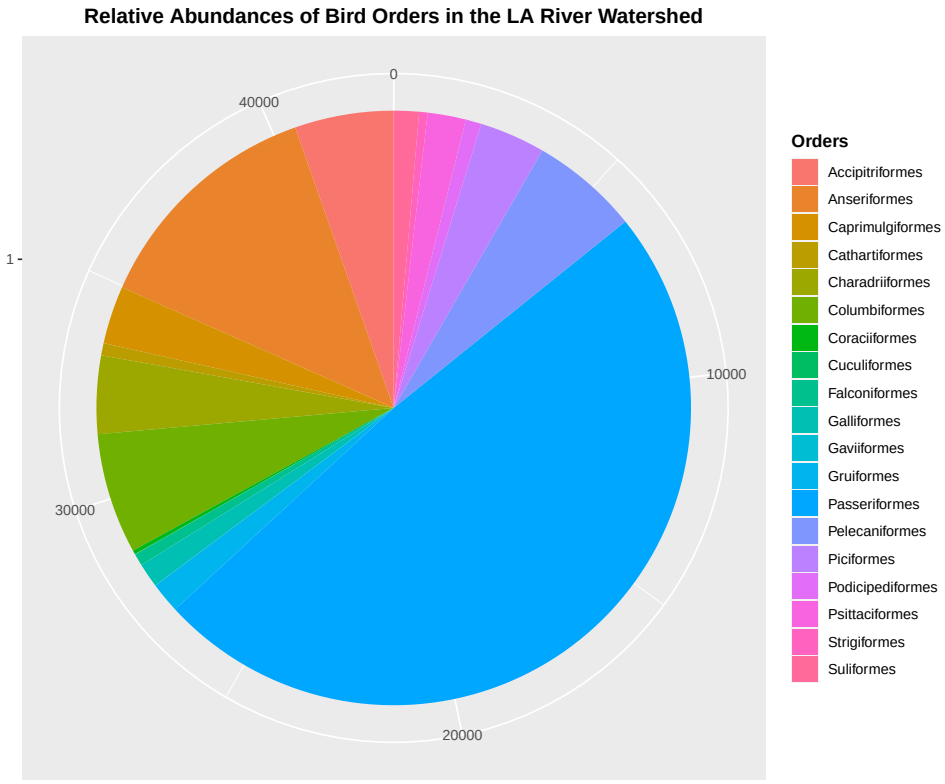

Figure 2: The relative abundances of bird orders within the LA River Watershed region based on iNaturalist data.