Sage Babish and Victoria Peechatt

BIOL792 Final Project

2023-12-06

NumPy, pandas, data cleaning and visualization

For this project, we used Pandas and NumPy to clean and extract data from our datasets. We both have rather different study systems and datasets, but we were able to make a couple programs useful for both of us (maybe with a little tweaking). Here, we present our problems and the steps we followed to address those problems, largely in the form of python code. We also present our next steps, both with our data and what else we hope to learn to do with NumPy, pandas, and python.

Sage has multiple people's specimen catalogs, data on snake TTX resistance in multiple formats, and a host of other unorganized files to extract data from.

I needed two different sets of information from these data sets. First, which specimens collected by our lab and our collaborators were from my study area, and did we have tissue samples, sequencing data, or phenotypes for them? Second, which specimens had phenotypes, base speeds, and at least one out of SVL, mass, and tail length? The first information was used to figure out how many tissue/genetic samples we already had and how many new ones would need to be requested from museums or captured in the field and phenotyped. The second set will be used to model the effects of mass, size, body condition, and TTX resistance on base speed in an attempt to observe a trade-off between resistance and muscle function.

Part 1: Catalog Searches

My first steps for this project were parsing through several specimen collection catalogs and extracting specimens based on increasingly specific qualifications. This is what the heads of some of those catalogs look like, to give an idea of what I was working with:

```
##   CollPreNo CollNo CollPre      Genus           Sp        Ssp      County
## 1  CRF 0001      1     CRF Thamnophis      elegans terrestris    Alameda
## 2  CRF 0002      2     CRF  Pituophis melanoleucus  catenifer Stanislaus
## 3  CRF 0003      3     CRF     Lanius ludovicianus
## 4  CRF 0004      4     CRF
## 5  CRF 0005      5     CRF
## 6  CRF 0006      6     CRF
##        State Latitude Longitude  DateColl Sex Remarks Stage
## 1 California                    20-Mar-95
## 2 California
## 3                                              DOR
## 4
## 5
## 6
##                                                                                 L
ocal
## 1 behind Lawerence Hall of Science, off Stadium Way, Berkeley Hills, Berkeley, Alameda C
o., CA
## 2
## 3
## 4
## 5
## 6
##        Collector.s. Alt..ID CollSpLocal Tissues Museum CatNo EcoData Country
## 1 C.R. Feldman, HWG                 NA                  NA             USA
## 2                                   NA                  NA             USA
## 3                                   NA                  NA             USA
## 4                                   NA                  NA             USA
## 5                                   NA                  NA             USA
## 6                                   NA                  NA             USA
##      Class OrderSub Family Continent Skel CollDateYear CollDateMonth
## 1 Reptilia                                       1995           Mar
## 2 Reptilia                                         NA
## 3                                                  NA
## 4                                                  NA
## 5                                                  NA
## 6                                                  NA
##   CollDateDay       Elev Tiss Pres IdDate Pubs GenBank
## 1          20 ca. 300 ft   NA               NA
## 2          NA              NA               NA
## 3          NA              NA               NA
## 4          NA              NA               NA
## 5          NA              NA               NA
## 6          NA              NA               NA
```

```
##           ID             species         locality    county sex capture_date
## 1 "Sneaky 2"       Charina botae                             -
## 2   "Sneaky"       Charina botae                             -
## 3        CRF      Charina bottae       Ash Canyon        NV   -
## 4    CRF2393  Thamnophis elegans McCullough Ranch    Sonoma   F
## 5    CRF2509      Charina bottae        Lily Lake El Dorado   -
## 6    CRF2980 Coluber constrictor                       Yolo   -
##   lab_entry_date SVL.mm._at_lab_entry mass.g._at_lab_entry mass.g._14Oct2017
## 1                                  NA                   NA                NA
## 2                                  NA                   NA                NA
## 3      27-Feb-17                   NA                   27                NA
## 4      12-May-13                   NA                   NA                NA
## 5      27-Jun-14                   NA                   90                NA
## 6      21-Apr-16                   NA                   NA                NA
##   SVL.mm._16April2018 mass.g._16April2018 SVL.mm._before_muscle_exp
## 1                  21                   6                        NA
## 2                  40                  41                        NA
## 3                                                                 NA
## 4                77.5                 255                        NA
## 5                   -                   -                        NA
## 6                63.5                  93                        NA
##   mass.g._before_muscle_exp food_type                  Ivermectin_treatment
## 1                        NA
## 2                        NA
## 3                        NA      mice 12-JULY-2017/27-JULY-2017/10-AUG-2017
## 4                        NA           12-JULY-2017/27-JULY-2017/10-AUG-2017
## 5                        NA      mice 12-JULY-2017/27-JULY-2017/10-AUG-2017
## 6                        NA      mice 12-JULY-2017/27-JULY-2017/10-AUG-2017
##                    Notes X50.MAMU genotype
## 1
## 2
## 3
## 4             Socialized
## 5 FOUND DEAD 14-DEC-2017
## 6
```

```
##   CollPreNo CollNo CollPre       Genus       Sp      Location County State
## 1    KER001      1     KER  Thamnophis  couchii    Deep Creek Fresno    CA
## 2    KER002      2     KER  Thamnophis  couchii    Deep Creek Fresno    CA
## 3    KER003      3     KER Thamnophis  sirtalis    Deep Creek Fresno    CA
## 4    KER004      4     KER Thamnophis   couchii Emerald Pools Nevada    CA
## 5    KER005      5     KER Thamnophis   couchii Emerald Pools Nevada    CA
## 6    KER006      6     KER Thamnophis   couchii Emerald Pools Nevada    CA
##   Latitude Longitude  DateColl TimeColl  MAMU Toxicity..mg.
## 1 36.93379 -119.2463  6/1/2021    12:55    NA            NA
## 2 36.93379 -119.2463  6/1/2021    13:05 60.00            NA
## 3 36.93379 -119.2463  6/2/2021     8:30  4.79            NA
## 4 36.31927 -120.6567 6/10/2021    12:00  1.01            NA
## 5 36.31927 -120.6567 6/10/2021    12:00  1.10            NA
## 6 36.31927 -120.6567 6/10/2021    13:30  0.41            NA
##                           Notes Stage  X X.1 X.2 X.3 X.4 X.5 X.6 X.7 X.8 X.9
## 1 stationary in water when found adult NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## 2                   on pond edge adult NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## 3            under rock, flipped       NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## 4                                      NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## 5                                      NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## 6                                      NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
##   X.10 X.11 X.12 X.13 X.14 X.15 X.16 X.17 X.18 X.19 X.20 X.21 X.22
## 1   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## 2   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## 3   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## 4   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## 5   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## 6   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
```

extract_couchii.py

The first value to extract specimens by was species, pulling out all the specimens that were *Th. couchii*, which I did with this script:

```
#!/usr/bin/env python3
import sys
import pandas as pd

for file in sys.argv[1:]:  #loop through all files passed to the script
  IN=pd.read_csv(file)    #read file in as a csv
  print(IN.head())        #print out the head mostly for reference/awareness of what files lo
ok like; also useful to see if species data is in a column named differently than will be cau
ght by the program

  if ("Sp" in list(IN.columns)): #most common column heading
      Couchii=IN.loc[IN["Sp"].str.strip() == "couchii"]
      Couchii.to_csv("couchii_mamu.csv",index = False,mode='a') #appending because I loop thr
ough multiple files

  elif ("Species" in list(IN.columns)):
      Couchii2=IN.loc[IN["Species"].str.strip() == "couchii"]
      Couchii2.to_csv("couchii_mamu.csv",index=False, mode='a') #output file name is hardcode
d but could easily be changed

  elif ("Spp" in list(IN.columns)):
      Couchii3=IN.loc[IN["Spp"].str.strip() == "couchii"]
      Couchii3.to_csv("couchii_mamu.csv",index=False, mode='a')

  IN.close() #not completely sure if you have to do this in pandas but it seems like best pra
ctices
```

This was the first script I made, and as you can see it was pretty hard-coded. To process multiple files at once, I needed that nested if-else loop because different files had different names for the column with the species name.

extractbycol.py

Next, I needed to extract all the specimens from specific counties (Sierra, Nevada, and Placer). I originally did this with another hard-coded script, but then realized I'd likely need to do something like this again and generalized it into a script that pulls out all the observations in a dataframe with a given value in a given column. It allows you to pull based on multiple values of interest, but only processes one file at a time because different files again had different column names and I wasn't sure how to divide up my `sys.argv` inputs.

```
#!/usr/bin/env python3
# only works on 1 file at a time
import sys, re
import pandas as pd

df=pd.read_csv(sys.argv[1]) #import input file
out=sys.argv[2] #save name of output file
col=sys.argv[3] #save column we're filtering by

for value in sys.argv[4:]: #loop through all inputted search terms
    fileout=df[df[col].str.contains(value, flags=re.IGNORECASE,na=False)] #pull   rows with t
he values passed as inputs in the column we specified
    fileout.to_csv(out,mode='a') #set to append because we loop through multiple values of in
terest
```

I liked being able to specify the name of the output file instead of having it based on the input file name - this isn't implemented in some of the scripts below but that's because they were made first and I didn't think of it at the time.

dropduplicates.py

At this point, I had a file with all the *couchii* specimens from 4 different specimen collections. The problem was, some of those specimens were listed in multiple specimen catalogs because they were collected by people in our lab and then sometimes put into my advisor's tissue sample catalog. So, I needed an easy way to remove duplicate observations from my data.

This script allows you to remove duplicates based on values in any column, partially because sometimes the "CollPreNo"/"Specimen ID" column is named differently and partially because sometimes you might only want one observation from each geographic region, or date, or some other metric.

```
#!/usr/bin/env python3
import pandas as pd
import sys

df=pd.read_csv(sys.argv[1]) #first input is the file to remove duplicates from
dup_check=str(sys.argv[2]) #second input is the column i want to check for duplicates
df=df.astype({dup_check:str}) #need to make sure that that column is read as a string so that
I can strip out/off any whitespace

df[dup_check]=df[dup_check].str.replace(" ","") #remove whitespace
uniq_df=df.drop_duplicates(subset=[dup_check],keep='first') #remove duplicates and keep the f
irst instance (could make that a separate argument if I wanted)
uniq_df.to_csv("noduplicates_"+sys.argv[1],index=False) #write to csv with a descriptvie pref
ix and the original name of the file
```

After this, I extracted all the specimens that we had phenotypes for (based on if their ID appeared in two files that listed almost all the phenotypes we had). I foolishly over-wrote that script when adapting it into something similar, so I can't show it here, but it worked similarly to extractbycol and the script below.

checkmamu.py (making sure there wasn't new data in new mamu files)

A little while after I did that, my advisor sent me more data files with phenotypes to confirm that there weren't any that didn't make it into the masterdoc. I built the below program to go through all those files and compare them with the list of specimens with phenotypes I already had and make sure I hadn't missed any. I did this by finding all the specimens in my document of *Th. couchii* from my study area that appeared in these phenotype files and then comparing them to my list of specimens we had phenotypes for ("specimens_to_find.csv" in the code below). Those that were missing had their IDs written to a text file. I manually copied over their information because the code was already kind of long and messy, and there were ultimately only 5 specimens missing.

```
#!/usr/bin/env python3
import pandas as pd
import sys

df=pd.read_csv(sys.argv[1],header=0) #read in the file to make the checks on
PreNo=str(sys.argv[2]) #give name of column with specimen ID info
df2=pd.read_csv(sys.argv[3],header=0) #read in file for the first check (in this context it w
as the file with all the snakes from my study area)
PreNo2=str(sys.argv[4]) #column name for the first check
df3=pd.read_csv("../specimens_to_find.csv",header=0) #read in file for the second check (hard
coded here to be whether the specimen is already on my list)
PreNo3="Collector #" #hard-coded column name for the second check

df[PreNo]=df[PreNo].str.replace(" ","") #strip white space
df2[PreNo2]=df2[PreNo2].str.replace(" ","") #strip white space
df=df.assign(ChkCounty=df[PreNo].isin(df2[PreNo2]).astype(int)) #column for if each entry is
in first vector
df=df.assign(ChkOld=df[PreNo].isin(df3[PreNo3]).astype(int)) #column for if each entry is in
second file

OUT=open("new_mamus.txt",mode='a') #open output txt file

for _, row in df.iterrows(): #very cursed way of doing this, I wouldn't do this now
    if row['ChkCounty'] == 1: #if it's in the first file
        if row['ChkOld']==0: #and not in the second file
            OUT.write(str(row[PreNo])+"\n") #add its ID to the output file

OUT.close() #close output file
```

This is not remotely how I would solve this problem anymore, but this was before learning more about pandas either on my own or in class, so it was the best I could come up with. I don't need this program anymore because shortly afterwards we decided to include samples that didn't have phenotypes, so I didn't bother updating it. I figure leaving it like this is good evidence of how much better I've gotten at python and pandas over the course of this project and this semester.

mamuchk.py

Once we knew we wanted specimens with and without phenotypes, and after adding another county to our study area, I needed to go back over my data and find a way to easily add a column that indicated if we did or did not have a phenotype. This program (with a name admittedly too close to the previous program), does that. It does have the phenotype files hard-coded in because I didn't (and still don't) anticipate using it for anything else, and I find programs with too many inputs difficult to keep track of and call correctly (see future goals

section for a proposed solution to this).

```
#!/usr/bin/env python3
import pandas as pd
import sys

df=pd.read_csv(sys.argv[1],header=0) #read in file to check for phenotypes
PreNo=str(sys.argv[2]) #column of specimen ID (or other thing to check using)
df2=pd.read_csv("old_MAMUs.csv",header=0) #hard-coded but could be adapted to arg
df3=pd.read_csv("KER_MAMU_101923.csv",header=0) #same as above

df[PreNo]=df[PreNo].str.replace(" ","") #strip white space from all 3
df2['ID']=df2['ID'].str.replace(" ","")
df3['ID']=df3['ID'].str.replace(" ","")

df=df.assign(ChkOld=df[PreNo].isin(df2['ID']).astype(int)) #similar to previous script, colum
n with value for if it was in first file
df=df.assign(ChkNew=df[PreNo].isin(df3['ID']).astype(int)) #as above, second file

df['MAMU'] = df['ChkOld'] #add a new column to the data frame by duplicating (odd way to do t
his in hindsight)
df['MAMU'] = "n" #set default value as n
df.loc[df['ChkOld'] == 1, 'MAMU'] = "y" #assign it y if in first file
df.loc[df['ChkNew'] == 1, 'MAMU'] = "y" #assign it y if in second file

df.to_csv(sys.argv[1]+"_mamudat.csv",index=False,mode='w') #output to file with suffix showin
g it's been modified by this program
```

Again, this was probably not the most efficient way to do this, especially with the creation of two extra columns, but it worked and that was good enough for my purposes at the time. If I were to do it again, I'd store the 'ChkOld' and 'ChkNew' values internally and not actually add them to the file, and make it less hard-coded so it's adaptable for other data (such as whether or not there's lat-long data, if we know what watershed it was, juvenile/adult, etc.)

The Output

At the end of all this (and some other manipulations I didn't show), I had a file with all the specimens of *Th. couchii* from my study area (El Dorado, Placer, Sierra, and Nevada counties) and a column showing whether or not we had a phenotype (MAMU) for each specimen. I also added columns for whether or not we have DNA extractions or DNA sequences for each specimen by a similar process; that code not shown here for the sake of space. The head of that file is shown here:

```
##     Museum.. Collector..  Lab..     Locality  Watershed County..State Latitude
## 1 no voucher     CRF2669 15.022 Canyon Creek Upper Yuba    Nevada, CA 39.44269
## 2 no voucher     CRF2670 15.023 Canyon Creek Upper Yuba    Nevada, CA 39.44269
## 3 no voucher     CRF2671 15.024 Canyon Creek Upper Yuba    Nevada, CA 39.44269
## 4 no voucher     CRF2672 15.025 Canyon Creek Upper Yuba    Nevada, CA 39.44269
## 5 no voucher     CRF2673 15.026 Canyon Creek Upper Yuba    Nevada, CA 39.44269
## 6 no voucher     CRF2674 15.027 Canyon Creek Upper Yuba    Nevada, CA 39.44269
##   Longitude MAMU Tissue. DNA. Seq. Notes..hallas.or.mine.
## 1 -120.6584    y       ~    y    n                crf dna
## 2 -120.6584    y       ~    y    y                    seq
## 3 -120.6584    y       ~    y    y                    seq
## 4 -120.6584    y       ~    y    n                crf dna
## 5 -120.6584    y       ~    y    n                crf dna
## 6 -120.6584    y       ~    y    y                    seq
```

Part 2: Compiling Old Data

My other big goal for this project was to go through some other, much older data files and extract all the relevant information while combining them into one file. This was inconvenient only because the files all had very mismatched columns, and because many of the entries were missing data I needed (so I didn't want the entries if they didn't have that data). It was overall a much less complicated process than the first goal, partially because I had gotten better about hard-coding everything and partially because I eventually gave up on my goal of using regular expressions to align similarly named columns (see future goals section).

Before doing anything, I had to run `drop_duplicates` on each input file. The files contained data of multiple trials for each individuals, so there were multiple lines for each individual. I only wanted the first line because that was the line with the phenotype, so I ran `drop_duplicates` as above. Here is the head of one of the files before and after removing extra observations:

```
##   INDIV  WT SVL  TL Trial1 Trial2  BASE      INJ.. TTX.DOSE.mg. POST.TIME
## 1     9 4.8 209 285  1.401  1.413 1.407   1(8MAMU)      0.00055     1.485
## 2     9 4.8 209 285  1.401  1.413 1.407  2(16MAMU)      0.00110     1.372
## 3     9 4.8 209 285  1.401  1.413 1.407  3(40MAMU)      0.00270     1.516
## 4     9 4.8 209 285  1.401  1.413 1.407 4(100MAMU)      0.00700     1.914
## 5     9 4.8 209 285  1.401  1.413 1.407 5(200MAMU)      0.01400     5.410
## 6    10 7.2 243 340  1.080  1.324 1.202   1(8MAMU)      0.00082     1.525
##   POST.BASE X1MAMU.snake  X  MAMU
## 1    94.75%  0.000068568 NA 149.5
## 2   102.55%  0.000068568 NA    NA
## 3    92.81%  0.000068568 NA    NA
## 4    73.51%  0.000068568 NA    NA
## 5    26.01%  0.000068568 NA    NA
## 6    78.82%  0.000102852 NA    NA
```

```
##   INDIV   WT SVL  TL Trial1 Trial2  BASE    INJ.. TTX.DOSE.mg. POST.TIME
## 1     9  4.8 209 285  1.401  1.413 1.407 1(8MAMU)      0.00055     1.485
## 2    10  7.2 243 340  1.080  1.324 1.202 1(8MAMU)      0.00082     1.525
## 3    11  7.5 222 301  1.451  1.475 1.463 1(8MAMU)      0.00086     1.386
## 4    12  5.1 228 314  1.158  1.010 1.084 1(8MAMU)      0.00058     1.091
## 5    13 16.8 372 499  0.677  0.904 0.791 1(8MAMU)      0.00190     0.995
## 6    14 41.4 477 650  0.707  0.904 0.806 1(8MAMU)      0.00470     1.162
##   POST.BASE X1MAMU.snake Unnamed..12  MAMU
## 1    94.75%  0.000068568          NA 149.5
## 2    78.82%  0.000102852          NA    NA
## 3   105.56%  0.000107138          NA  63.5
## 4    99.36%  0.000072850          NA  36.5
## 5    79.45%  0.000239988          NA  61.5
## 6    69.32%  0.000591399          NA 123.8
```

And, for the sake of comparison, here are the heads of a couple other files to show just how differently they were all organized.

```
##   INDIV species Locality    WT SVL  TL Trial1 Trial2  BASE INJ.. TTX.DOSE
## 1  3.30    T.c.  Hat Cr. 129.5 700 856  0.545  0.647 0.596 0.5MU  0.00093
## 2  3.31    T.c.  Hat Cr. 126.3 666 844  0.575  0.688 0.632 0.5MU  0.00090
## 3  3.32    T.c.  Hat Cr.  98.7 630 811  0.631  0.660 0.646 0.5MU  0.00071
## 4  3.33    T.c.  Hat Cr.  91.6 656 836  0.588  0.513 0.551 0.5MU  0.00065
## 5  3.34    T.c. Deer Cr.  82.4 649 830  0.609  0.666 0.638 0.5MU  0.00059
## 6  3.35    T.c. Deer Cr.  98.5 600 772  0.582  0.643 0.613 0.5MU  0.00070
##   POST.TIME POST.BASE ml.of.Dilution dilution Unnamed..15 MAMU
## 1     0.917    64.99%          0.093 .01mg/ml          NA 1.71
## 2     0.881    71.68%          0.090 .01mg/ml          NA 2.21
## 3     0.764    84.49%          0.071 .01mg/ml          NA 1.42
## 4     0.601    91.60%          0.065 .01mg/ml          NA 1.44
## 5     0.628   101.51%          0.059 .01mg/ml          NA  1.7
## 6     0.678    90.34%          0.070 .01mg/ml          NA 2.19
```

```
##    INDIV Collector.ID    Species Locality         Unnamed..4   WT SVL TL Trial1
## 1 16.100      EJE 158 T. couchii   Nevada Faucherie Lake Dam 56.8  NA NA  0.946
## 2 16.101      EJE 159 T. couchii   Nevada Faucherie Lake Dam 33.2  NA NA  0.663
## 3 16.102      EJE 160 T. couchii   Nevada Faucherie Lake Dam  9.4  NA NA  0.913
## 4 16.108     CRF 3063 T. couchii Toulumne       Cherry Creek 89.2  NA NA  0.865
## 5 16.109     CRF 3064 T. couchii Toulumne       Cherry Creek 48.8  NA NA  0.706
## 6 16.110     CRF 3065 T. couchii Toulumne       Cherry Creek 59.4  NA NA  0.702
##   Trial2  BASE  INJ.. TTX.DOSE.mg. POST.TIME POST.BASE X1.MU.g ml.of.Dilution
## 1  0.720 0.833  5MAMU   0.00405836     1.301    64.03%      NA          0.081
## 2  0.756  0.71  5MAMU   0.00237214     1.139    62.29%      NA          0.047
## 3     NA 0.913  5MAMU   0.00067163     1.889    48.33%      NA          0.013
## 4     NA 0.865 25MAMU   0.03186670        NA   #DIV/0!      NA          0.319
## 5  0.796 0.751 25MAMU   0.01743380     1.183    63.48%      NA          0.174
## 6  0.867 0.785 25MAMU   0.02122065     1.023    76.69%      NA          0.212
##   dilution       MAMU
## 1     0.05        8.6
## 2     0.05       21.4
## 3     0.05       21.7
## 4     0.10 Gave Birth
## 5     0.10         69
## 6     0.10       85.3
```

Once that was done, I was ready to concatenate and filter my files.

concat_keepsetcols.py

I actually did this all in one program (yay efficiency!), which takes a few arguments: the number of files you're passing to it, the files, and the names of the columns you want to keep for the final output. That was important because there was a lot of information in these files I didn't need in my output file, and it was easier to say which to keep than manually delete all those extra columns.

```
#!/usr/bin/env python3
import sys
import pandas as pd
import numpy as np

concat_df = pd.DataFrame() #initialize an empty df to combine everything into

numfiles=int(sys.argv[1]) #first argument if number of files
fileend=numfiles+2 #and that is used to figure out how to index through sys.argv
for file in sys.argv[2:fileend]: #loop through every file
    df=pd.read_csv(file,dtype=str) #read in file as data frame
    concat_df = pd.concat([concat_df,df],ignore_index=True) #concatenate by column, adding ne
w columns every time a new one is introduced, and putting NA for any entries that don't have
a vzlue for that column

out_df=pd.DataFrame() #initialize another df for actual output
for col in sys.argv[fileend:]: #loop through all column names passed as arguments
    out_df = pd.concat([out_df,concat_df[col]],axis=1) #add on the columns specified

out_df[sys.argv[fileend]]=out_df[sys.argv[fileend]].replace('[^\d\.]',np.nan,regex=True) #thi
s was skirting the edge of hard-coding, it removed everything except numbers and decimal poin
ts. Though in this case all the data I kept was numeric, I did it for a specific column becau
se if I were to do it for the entire data frame, any text data would automatically be lost. I
t replaces the strings and characters with an NA value instead of blank space, which allows t
hem to be stripped out by the below.

out_df.dropna(subset=[sys.argv[fileend],sys.argv[fileend+1]], inplace = True) #remove rows th
at don't have entries for at least the first two columns

out_df.to_csv("concatenatedcols.csv",index=False) #output! hard-coded name because there were
too many arguments already
```

Ultimately, I ended up with the file below, which had the data I needed on each snake and nothing extra. As you can see, there are no NAs in the first two columns because I specified those as the two variables that needed to be present for an entry to be worth including.

```
##    MAMU  BASE   WT SVL  TL
## 1 149.5 1.407  4.8 209 285
## 2  63.5 1.463  7.5 222 301
## 3  36.5 1.084  5.1 228 314
## 4  61.5 0.791 16.8 372 499
## 5 123.8 0.806 41.4 477 650
## 6 119.5 0.688 23.3 384 532
```

```
##       MAMU             BASE              WT               SVL
##  Min.   :  0.10   Min.   :0.4640   Min.   :  2.500   Min.   :175.0
##  1st Qu.:  4.05   1st Qu.:0.7133   1st Qu.:  4.165   1st Qu.:195.0
##  Median : 24.05   Median :0.8785   Median :  9.250   Median :261.0
##  Mean   : 41.36   Mean   :0.9179   Mean   : 34.326   Mean   :360.9
##  3rd Qu.: 74.85   3rd Qu.:1.1180   3rd Qu.: 44.775   3rd Qu.:507.2
##  Max.   :161.50   Max.   :1.5310   Max.   :252.700   Max.   :854.0
##                                    NA's   :54        NA's   :60
##        TL
##  Min.   : 232.0
##  1st Qu.: 258.2
##  Median : 295.5
##  Mean   : 446.3
##  3rd Qu.: 628.5
##  Max.   :1100.0
##  NA's   :90
```

This data frame will be used for future visualization and modeling (below), but this was the extent of the pandas-based manipulation I wanted to do on these files!

Future Goals

This was all I did with this data for this project, but there's some future goals I have for both my python scripts and my data that fell either outside the theoretical scope of this project or were beyond my ability to tackle at this point in time. Here are some of the ones I mentioned throughout the write-up, with a little more detail about each:

- Input Prompts in Scripts - Python has a variety of ways to incorporate input from the user into the operation of a script, including the `cmd` module and `input()` (a built-in function). Some of these programs would have really benefitted from this, and I wouldn't have felt the need to hard-code quite so much if I knew how to do this. My last script is a pretty good example of an overly complicated amount and order of command-line inputs, and I think running that program would be a little more bearable if I was prompted for the files, values, and column names one at a time.
- Regular Expression Column Matching for Concatenating - I spent a good amount of time on stack exchange trying to figure out how to concatenate data frames according to columns with similar names. For example, "ID" and "INDIV" are pretty close, and so are "Sp" and "Species" (though the latter is much more likely to work with regular expressions in my opinion). Some of the mismatches I doubt could be resolved this way ("WT" and "Mass" have absolutely nothing in common), and changing them by hand is probably more efficient than a program at this point given the relatively small number of files I have, but it's something I might consider in the future (or, you know, just making sure that my personal files are never this unorganized).
- Data Visualization - Peechatt got into this some in her data manipulation, but I spent too long on all the scripts I was making to be able to really do it justice (and I just feel better plotting things in R). In the long term I definitely plan to make some correlation plots with my dataset from part 2 and see what relationships may or may not be there. I'll definitely be adapting her code to do it though!
- Modeling in Python? - The question mark is there not because I don't think it's possible but because I have no idea how to go about it. Also with my dataset from part 2, I want to make a model investigating the relationship between base speed and the other 4 variables. I have a good idea of how to do this in R in several different ways, but I have no clue how to do it in python. Realistically I will probably end up

doing this in R because I'm so much more comfortable with it, but I'd like to spend some time at least reading up on and messing around with simpler models in R.

Peechatt has three 20+ year datasets with caterpillar, host plant, and other probably interesting data.

- First, I imported the datasets, cleaned the data, and extracted the columns I wanted to look at further.
- I've used the data before to calculate the host breadth of the species that were collected. Previously, I manually went through the excel, sorted by lep species name, and counted the unique number of host plants. But with pandas and numpy, it took 4 lines of code!
- I used matplotlib to make figures of species richness, rank abundance, etc. for the sites.

Import dataset, and clean! although your data cleaning is really dependent on what you have to work with.

```
#!/usr/bin/env python3

import sys, re, glob
import numpy as np
import pandas as pd

import matplotlib
matplotlib.use
import matplotlib.pyplot as plt
plt.style.use('ggplot')

df = pd.read_csv("Costa_Rica_Database_2022.csv",
                 usecols=['ID','Date Collected', 'locale',
                          'plant family', 'plant species', 'plant common name',
                          'order','family','sub family','lep species','lep name'],
                parse_dates=['Date Collected'],infer_datetime_format=True)

#Cleaning up locale names so that they are the first three characters only
df['plot_s']=df['locale'].str[:3]
```

Locating a lep

Using Sage's script ("extract_couchii.py"), I was able to extract information for a lep species of interest. I customized their script to suit my dataset - it took about a minute to use.

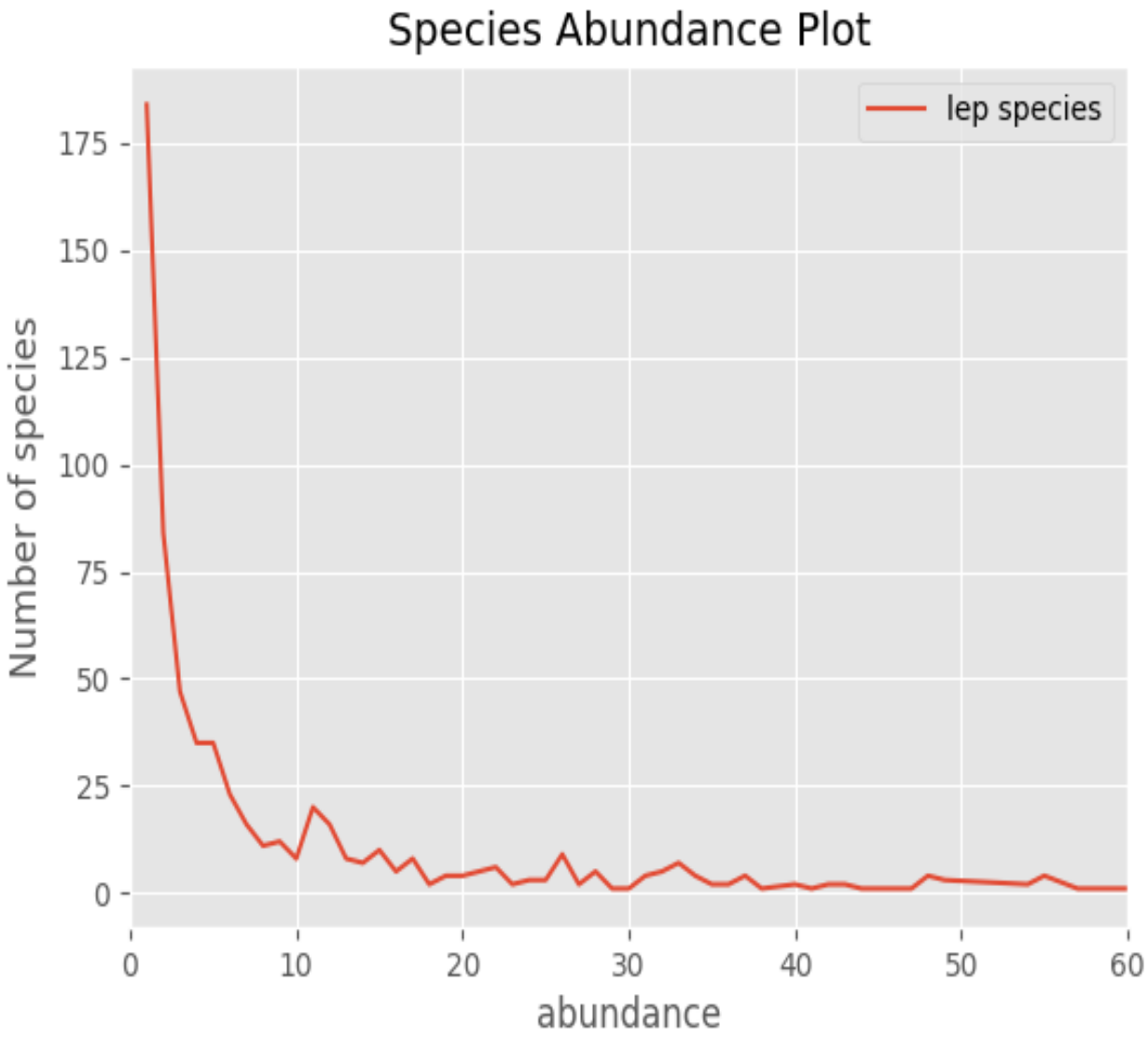
```
for file in sys.argv[1:]:  #loop through all files passed to the script
  IN=pd.read_csv(file)

  if ("lep species" in list(IN.columns)): #most common column heading
      Couchii=IN.loc[IN["lep species"].str.strip() == "quadrus cerialis"]
      Couchii.to_csv("quadceri.csv",index = False,mode='a')

  elif ("Lep Species" in list(IN.columns)):
      Couchii2=IN.loc[IN["Species"].str.strip() == "quadrus cerialis"]
      Couchii2.to_csv("quadceri.csv",index=False, mode='a')

  elif ("Lep species" in list(IN.columns)):
      Couchii3=IN.loc[IN["Spp"].str.strip() == "quadrus cerialis"]
      Couchii3.to_csv("quadceri.csv",index=False, mode='a')
```

```
##        ID Date.Collected     locale plant.family plant.species
## 1   307.2      12-Oct-97        str   piperaceae      piper sp
## 2 307.303      15-Oct-97   str 2800   piperaceae      piper sp
## 3   410.0       4-Jan-98    huertos   piperaceae piper auritum
## 4   687.0      26-May-98    huertos   piperaceae      piper sp
## 5   877.0      17-Jun-98 glasnost 2   piperaceae piper auritum
## 6   881.0      17-Jun-98 glasnost 2   piperaceae piper auritum
##   plant.common.name       order      family sub.family      lep.species
## 1                   lepidoptera hesperiidae  pyriginae quadrus cerialis
## 2                   lepidoptera hesperiidae  pyriginae quadrus cerialis
## 3                   lepidoptera hesperiidae  pyriginae quadrus cerialis
## 4                   lepidoptera hesperiidae  pyriginae quadrus cerialis
## 5                   lepidoptera hesperiidae  pyriginae quadrus cerialis
## 6                   lepidoptera hesperiidae  pyriginae quadrus cerialis
##          lep.name
## 1 piper hesperiid
## 2 piper hesperiid
## 3
## 4
## 5 piper hesperiid
## 6 piper hesperiid
```

Species Richness and Abundance Plots

Diversity indices are often scrutinized for not capturing the whole story, but this is my attempt to understand what exactly they are and how they can be visualized. My next steps would be to understand, code, and develop visualizations for other measures of diversity like beta and gamma diversity, interaction diversity, etc. This is just the beginning…

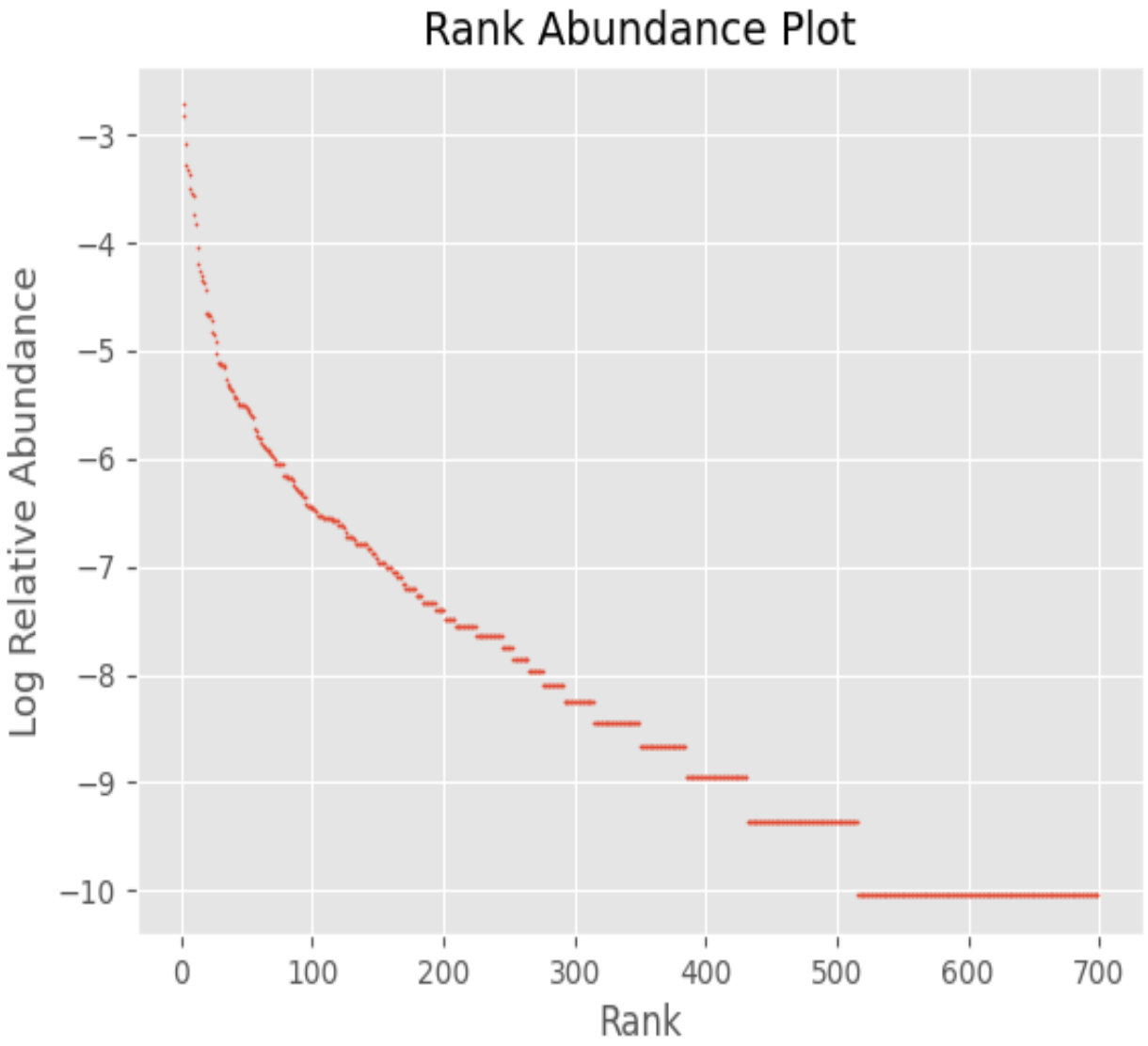
```
#Species Richness: number of species observed
richness = df['lep species'].nunique()
print("\nNumber of species: ", richness,"\n")

#Number of species vs. #of individuals observed
plt.figure()
spec_abun = df.groupby('lep species')['lep species'].count().rename('abundance')
spec_abun.to_csv("Abundance.csv")
df2 = pd.read_csv("Abundance.csv")
spec_numabun = df2.groupby('abundance').count().plot()
plt.xlim(0,60)
plt.ylabel("Number of species")
plt.savefig('Species_vs_Abundance.png')
```



Rank Abundance Plots

These are another great way to visualize diversity. Again, rare species are more commonly found. Common species are rarelky found. Higher ranks are given to species with rare abundances.

```
# Rank abundance plot : Relative abundancy vs. Rank (low rank is most abundant)
plt.figure()
spec_rank = df2.sort_values('abundance', ascending=False)
print("\nNumber of individuals: ", spec_rank['abundance'].sum(), "\n")
spec_rank['abunprop'] = spec_rank['abundance']/(spec_rank['abundance'].sum())
spec_rank['rank'] = spec_rank.reset_index().index
spec_rank = spec_rank.astype({'abunprop': float, 'rank':int})
spec_rank['rank'] +=1
spec_rank['logabun'] = np.log(spec_rank['abunprop'])
print(spec_rank)
plt.scatter(spec_rank['rank'], spec_rank['logabun'], s = 0.5)
plt.savefig('Rank Abundance.png')
```



Diet Breadth data (what I was after!)

One of my research interests is the evolution of specialization; we observe countless instances of specialization in host plant-caterpillar-parasitoid systems. I would love to continue the work documenting the interactions in these systems and give power to the data that people hve collected long term. Here's an attempt to visualize

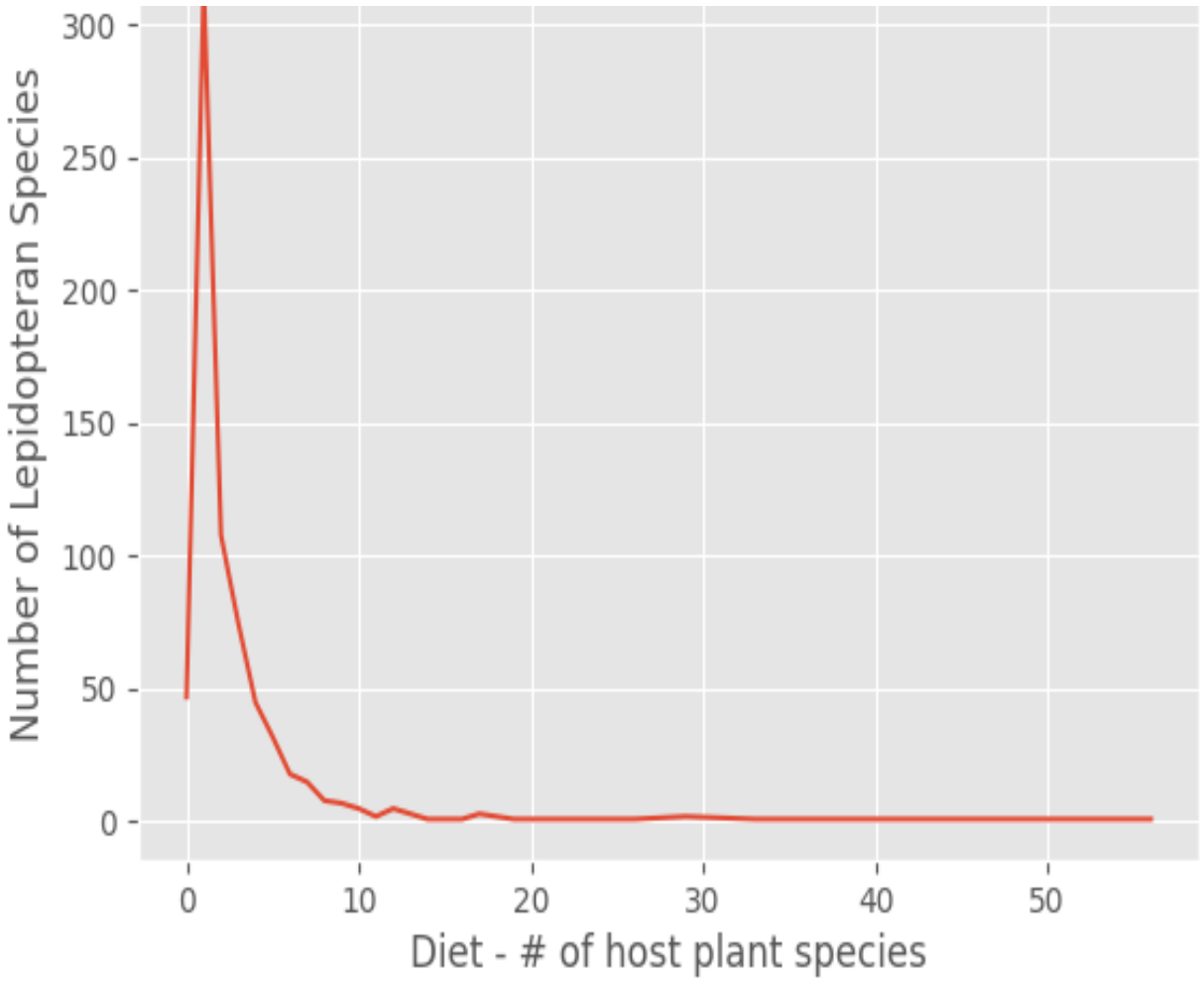
host plant specialization.

```
#Diet breadth, all instances of host plants recorded for each species
diet_names = df.groupby('lep species')['plant species'].unique()
diet_names.to_csv("names_diet.csv")
diet_number = df.groupby('lep species')['plant species'].nunique()
diet_number.to_csv("numbers_diet.csv")
df_names = pd.read_csv("names_diet.csv")
df_num = pd.read_csv("numbers_diet.csv")
diet_full = pd.merge(df_names, df_num, on='lep species')
diet_full.to_csv("Diet Breadth Summary.csv")
```

```
diet_breadth_num = read.csv("VPscripts/Diet Breadth Summary.csv")
head(diet_breadth_num)
```

```
##   X       lep.species plant.species_x
## 1 0   acharia horrida               1
## 2 1 acharia hyperoche               2
## 3 2     acharia nesea               8
## 4 3 acharia ophelians               1
## 5 4    acharia sarans               8
## 6 5        acharia sp               3
##
plant.species_y
## 1
['calathea crotalifera']
## 2
['musa acuminata' nan 'solanum sp']
## 3 ['anthurium sp' nan 'hyeronima alchorneoides' 'inga edulis' 'heliconia sp'\n 'anthurium
clavigerum' 'piper reticulatum' 'guatteria diospyroides'\n 'adelia triloba']
## 4
['heliconia sp']
## 5               [nan 'alchornea costaricensis' 'musa sp' 'heliconia sp'\n 'anthurium clavi
gerum' 'neea psychotroides' 'calathea lutea'\n 'anthurium sp' 'calathea sp']
## 6
['piper colonense' 'piper sp' 'heliconia sp']
```

```
plt.figure()
breadth = df_num.groupby('plant species')['lep species'].count().plot()
plt.xlabel("Diet - # of host plant species")
plt.ylabel("Number of Lepidopteran Species")
plt.title("Diet Breadth Plot")
plt.savefig("Diet Breadth Plot")
```



Future Goals

- Visualizing beta, gamma, interaction diversity
- Importing virus data to observe infection rates across taxa
- Importing parasitoid data to observe parasitism rates across taxa
- Combining data from 3 sites to show latitudinal, or at least geographic, variation