Background/Overview

In the summer of 2016, 92 different Pokémon GO players self-reported their primary location (location string and longitude/latitude) and the number of spawns of each species that they had observed. This problem set will walk through ways to describe these data as well as explore the biodiversity present within them. While the data collection strategy is somewhat different than one may experience in a standard ecology course, the process of data exploration, calculating descriptive statistics and computing measures of biodiversity are quite similar independent of how the samples were taken.

While you are writing your code, make sure to include comments so that you will remember why you are doing what you are doing!

Section 1: Importing the data

1.1 Data format

Use a text editor to explore the .csv file and see if you can determine what the structure of the file is.

1.1.1 Please briefly describe the file structure.

**** The top three lines of the file are header lines that describe the latitude, longitude and a descriptive location string for each of 92 participants, comma separated. The first column of the remaining lines starts with name of a Pokemon species followed by a comma separated list of the number of spawns for that species and that participant. ****

1.2 Import data

Using what you know about importing data into MATLAB, import it into the workspace. The best solution would be to generate some code that would be reusable for processing additional datasets in the future.

```
% Prompt the user to select a file
[csvFileName,csvFilePath] = uigetfile('*.csv',...
    'Please select the Pokémon GO datafile');

% open the file for reading
[fidA] = fopen([csvFilePath,filesep,csvFileName],'r+');

% The first three lines are headers of the latitude,
% longitude and string location name for the users who
% report their spawn rates. Each line has data
% separated by commas
tempLine = fgetl(fidA);
tempLineSplit = strsplit(tempLine,',');
if not(strcmp('latitude',tempLineSplit{1}))
    % formatting issue
    warning('PokemonGoDistributions:ReadingFormat',...
        'Selected file does not match the anticipated format.')
```

```
end

latitudeList = str2double(tempLineSplit(2:end));
nParticipants = length(latitudeList);

tempLine = fgetl(fidA);
tempLineSplit = strsplit(tempLine,',');
if not(strcmp('longitude',tempLineSplit{1}))
    % formatting issue
    warning('PokemonGoDistributions:ReadingFormat',...
        'Selected file does not match the anticipated format.')
end

longitudeList = str2double(tempLineSplit(2:end));

% location strings sometimes have internal commas and
% are therefore sometimes comma separated and sometimes
% comma and quote separated
tempLine = fgetl(fidA);
locationList = strsplit(tempLine,{'"'});
locationList(strcmp(locationList,',')) = [];

if isempty(locationList{end})
    locationList(end) = [];
end

% remove opening and closing commas
for iiLocation=1:size(locationList,2)
    if strcmp(locationList{iiLocation}(end),',')
        locationList{iiLocation}(end) = [];
    end
    if strcmp(locationList{iiLocation}(1),',')
        locationList{iiLocation}(1) = [];
    end
end

% first entry is a title for the list of pokemon
if strcmp(locationList{1},'Pokemon')
    locationList(1) = [];

end

% The remaining lines have the number of each pokemon
% spawns observed by each participant with the first
% element being the name of the Pokemon
counter = 0;
while not(feof(fidA))
    tempLine = fgetl(fidA);
    tempLineSplit = strsplit(tempLine,',','CollapseDelimiters',false);

    counter = counter+1;
    pokemonName(counter) = tempLineSplit(1);
    nSpawnsObserved(counter,:) = str2double(tempLineSplit(2:end));
```

```
end
```

Section 2: Accessing the dataset

When working with datasets that describe distributions of events, it can be tricky to know whether one should treat the data as counts of events, list of events, or probability of events. As we go through this problem set, we will see some of the benefits of each of these notions.

2.1 Counts of events

This is the format that our original dataset contains. For example, participant number 6 has (“56 Bulbasaur, 24 Charmander, 18 Squirtle”). Make sure that you’re comfortable pulling out the data associated with participant number 6, or any other specific participant.

2.1.1 How many Bulbasaur does participant 43 have?

```
bulbasaurIndex = strcmp(pokemonName,'Bulbasaur');
disp([nSpawnsObserved(bulbasaurIndex,43)])
```

```
    38
```

**** 38 ****

2.1.2 How many total Pikachu have participants numbered 40 to 50 seen?

```
pikachuIndex = strcmp(pokemonName,'Pikachu');
disp([sum(nSpawnsObserved(pikachuIndex,40:50))])
```

```
   102
```

**** 102 ****

2.2 List of events

The second format has one entry for each item as opposed to each species. So this list for participant number 60 would be (Bulbusaur, Bulbusaur, Bulbusaur, Charmander, Charmander, Squirtle, Squirtle, Squirtle, Squirtle, ...).

```
% Start by writing a snippet of code that takes a
% specific participant index and loops over each
% of the possible species.
iiParticipant = 60; % or another integer

participantSampleList = cell(0);
for iiSpecies = 1:size(nSpawnsObserved,1)
    % For each of these species, append the appropriate
    % number of that species to the end of a growing
    % list. You may choose to use cell arrays with
```

```
    % characters, string arrays, categorical data or simply
    % the numeric index into the list of species.
    participantSampleList = cat(1,participantSampleList,...
        repmat(pokemonName(iiSpecies),[nSpawnsObserved(iiSpecies,iiParticipant),1]));
end
```

2.3: Frequency of events.

When we want to compare two different distributions with different total numbers of events, it can sometimes be helpful to have a list of the frequency of occurrences in addition to the number of occurrences. For participant 60, this would look like (0.21% Bulbusaur, 0.14% Charmander, 0.28% Squirtle, ...). Create a third list of the percentages for each specific participant by dividing the counts observed by that participant by the total number of events observed by that participant.

NOTE: Be sure to use *nansum* instead of *sum* to add up the number of events in case there were non-numeric measurements imported.

```
percentageSpawnsObserved = nSpawnsObserved./(nansum(nSpawnsObserved,1));

% INSTRUCTOR NOTES: on earlier versions of matlab, this
% element-wise division would error with a size mismatch
% and the bsxfun notation could be substituted
%
% E.g.: percentSpawnObs = bsxfun(@rdivide,nSpawnsObserved,(sum(nSpawnsObserved,1)));
```

Section 3: Describing the *spawn counts* distribution?

Our next job is to describe how frequently we have seen each Pokémon. For this question, we don't care what the species is, just that each species is unique. We will use the data from participant 60 to start. We'll treat each species as a *measurement*.

3.1 Remove zero values

For some measurements, a value of *zero* is an actual measurement. In the case of the spawn data, it is not clear if the zeros are true measurements or not. For example, because there were not any *Grumpigs* in Pokémon GO during the time of this data collection, it would not make much sense to include that as a species and put a 0 in for it. So, generate a slightly modified dataset where you have removed all of the 0's and set them to a value of *nan*. NaN stands for Not a Number, a numeric data type that has no value.

```
% create a new variable with the zeros replaced with nans
nSpawnsObservedNans = nSpawnsObserved;
nSpawnsObservedNans(nSpawnsObserved==0) = nan;

% INSTRUCTOR NOTE: Students have a tendency to make this new
% variable but then forget to use it in future calculations.
```

3.2 Central tendency

One description we would like to have is something about the scale or typical number of times we have seen each type. The first measure of central tendency is the *mean* or average number of times a species was seen.

3.2.1 What is the *mean* value for the distribution of spawns from participant 60?

```
% Calculate the mean, the average number of times
% participant 60 saw each species.
meanVal = nanmean(nSpawnsObservedNans(:,iiParticipant));

disp(meanVal);
```

```
    23.6333
```

**** 23.633 ****

3.2.2 Plot a histogram of the data from participant 60 using 100 bins and add a vertical line at the mean value in red.

```
figure();
% turn hold on so that we can plot multiple things on the same graph
hold on;
% plot the histogram
histogram(nSpawnsObservedNans(:,iiParticipant),100);
% plot the vertical line
plot([meanVal,meanVal],ylim,'r','LineWidth',2);
% label the axes
xlabel('Number of times a species spawned');
ylabel('Number of species');
```

3.2.3 Out of the 73 species, how many of them were seen more frequently than the mean?

```
numberGreaterThanMean = nnz(nSpawnsObservedNans(:,iiParticipant)>meanVal);
disp(numberGreaterThanMean);
```

```
     9
```

**** 9 ****

3.2.4 Does the *mean* value accurately describe this center of this particular distribution?

**** No, the mean value graphically appears to be far above the center of this distribution. ****

3.2.5 The second measure of central tendency is the *median*. Calculate the *median* of the data. Some statisticians refer to the *mean* as the “average value” and the *median* as the “typical value”.

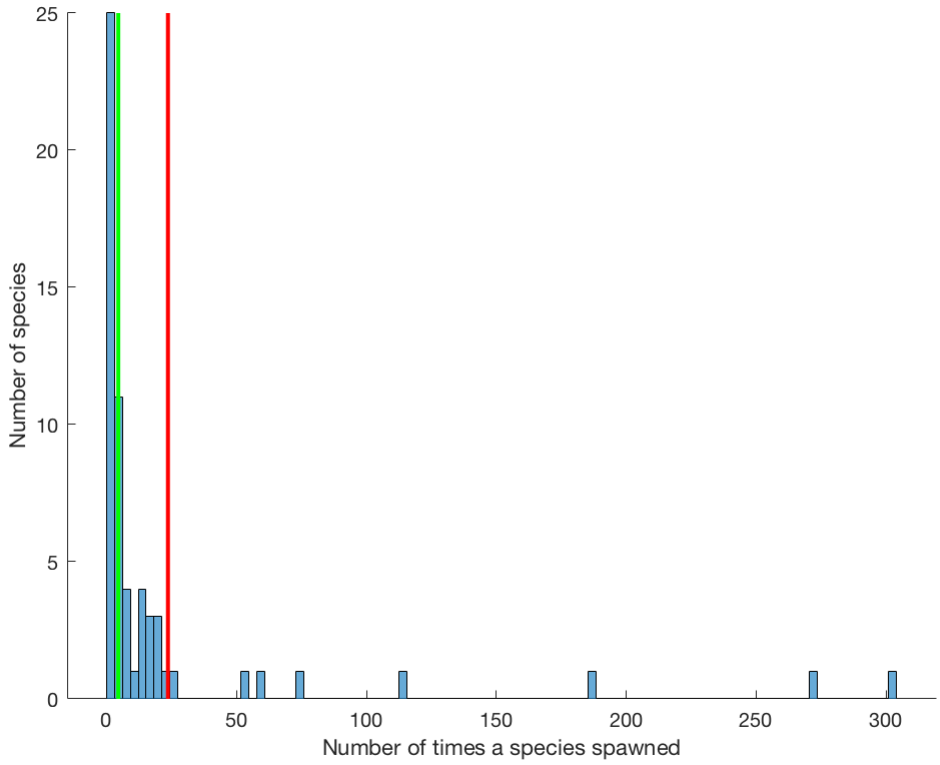
```
medianVal = nanmedian(nSpawnsObservedNans(:,iiParticipant));
disp(medianVal)
```

```
    4.5000
```

**** 4.5 ****

3.2.6 Plot the median on the histogram in a green vertical line.

```
plot([medianVal,medianVal],ylim,'g','LineWidth',2);
```




```
```

3.2.7 Calculate each of these again after removing the data for Pidgeys.

```
% ignoring pidgeys
isNotPidgey= not(strcmp(pokemonName,{'Pidgey'}));
meanWithoutPidgeys = nanmean(nSpawnsObservedNans(isNotPidgey,iiParticipant));
disp(meanWithoutPidgeys)
```

```
    18.8814
```

```
medianWithoutPidgeys = nanmedian(...
    nSpawnsObservedNans(isNotPidgey,iiParticipant));
disp(medianWithoutPidgeys)
```

```
     4
```

**** mean without Pidgeys, 18.8814

median without Pidgeys, 4 ****

3.2.8 Is the median or mean more influenced by extreme values?

**** Mean is much more sensitive to the extreme values ****

3.3 Spread

As we've already seen from the histogram, participant 60 did not see every species with exactly the same frequency. We have multiple measures of spread that we can use to help describe the variability between our *measurements*.

3.3.1 Start by calculating the *standard deviation* of counts for participant 60.

```
standardDeviationVal = nanstd(nSpawnsObservedNans(:,iiParticipant));
disp(standardDeviationVal);
```

```
    57.8809
```

**** 57.8809 ****

Another measure of spread is the *inter-quartile range*. This is calculated as the difference between the 75th percentile and the 25th percentile. In MATLAB, you can use the function **iqr** or the **quantile** function quantile(x,0.75)-quantile(x,0.25).

3.3.2 What is the *inter-quartile range* for participant 60?

```
iqrVal = iqr(nSpawnsObservedNans(:,iiParticipant));
disp(iqrVal);
```

```
    13.5000
```

**** 13.5 ****

3.3.3 Calculate the *standard deviation* and *IQR* again for the dataset without Pidgeys.

```
standardDeviationValNoPidgeys = nanstd(...
    nSpawnsObservedNans(isNotPidgey,iiParticipant));
disp(standardDeviationValNoPidgeys)
```

```
    45.0525
```

```
iqrValNoPidgeys = iqr(nSpawnsObservedNans(isNotPidgey,iiParticipant));
disp(iqrValNoPidgeys)
```

```
    12.7500
```

**** standard deviation without Pidgeys, 45.0525

iqr without Pidgeys, 12.75 ****

Section 4: Measuring the shape of the distribution.

One reason that many people report the *mean* and *standard deviation* of their measurements is that these are exactly the two parameters needed to describe a *normal* distribution, also known as a *Gaussian* distribution. However, when the data are not *normally* distributed, the *mean* and *standard deviation* are not the best way to describe the shape. One can either find a transformation that makes the data *normal*, or find the appropriate shape of the distribution. We will the use of the Lilliefors statistical test to compare transformed data to *normal* distributions.

4.1 Test for normality

The Lilliefors test asks the question, “are the data inconsistent with a normal distribution?” It can be excetued in MATLAB with [H,p] = lillietest(x); H is *true* or 1 if the null hypothesis that the data are normally distributed is rejected in favor of the alternate hypothesis that the data have a distribution with a shape that is something other than *normal.* The second output argument, p, is the p-value for rejecting the null hypothesis. p-values that are closer to 0 imply higher confidence in rejecting the null hypothesis.

4.1.1 Are the data for participant 60 normally distributed at an alpha value of 0.1?

```
% set alpha value for all tests
alphaVal = 0.01;

[H,p] = lillietest(nSpawnsObservedNans(:,iiParticipant),alphaVal);
```

```
Warning: P is less than the smallest tabulated value, returning 0.001.
```

```
disp(H);
```

```
     1
```

**** No, we reject the null hypothesis that the data are normally distributed. ****

Now let’s try some transformations of the data to see if we can find a simple transformation that makes them appear *normal*. If *x* is not normally distributed, try *x*^2, 1/*x*, and log(*x*).

4.1.2 Do any of these transformations fail to reject the null hypothesis at an alpha value of 0.01?

```
% try x^2
[H,p] = lillietest(nSpawnsObservedNans(:,iiParticipant).^2,alphaVal);
```

```
Warning: P is less than the smallest tabulated value, returning 0.001.
```

```
disp(['x^2: ',num2str(H)]);
```

```
x^2: 1
```

```
% try 1/x
[H,p] = lillietest(1./nSpawnsObservedNans(:,iiParticipant),alphaVal);
```

```
Warning: P is less than the smallest tabulated value, returning 0.001.
```

```
disp(['1/x: ',num2str(H)]);
```

```
1/x: 1
```

```
% try log(x)
[H,p] = lillietest(log(nSpawnsObservedNans(:,iiParticipant)),alphaVal);
disp(['log(x): ',num2str(H)]);
```

```
log(x): 0
```

**** Yes, we fail to reject the null hypothesis for the log transformed data. ****

If so, then we have an appropriate transformation for our data. We can compute the shape of the distribution in this transformed space and then undo the transformation. Find the *mean* value in the transformed space and then undo the transformation and plot that value on your histogram. For example, if the data appear normal after the transformation y = 1/x, calculate the *mean* value of y and plot 1/y_mean on your histogram. If the data appear *normal* after a log transform, calculate the *mean* value of y = log(x) and then plot exp(y_mean).

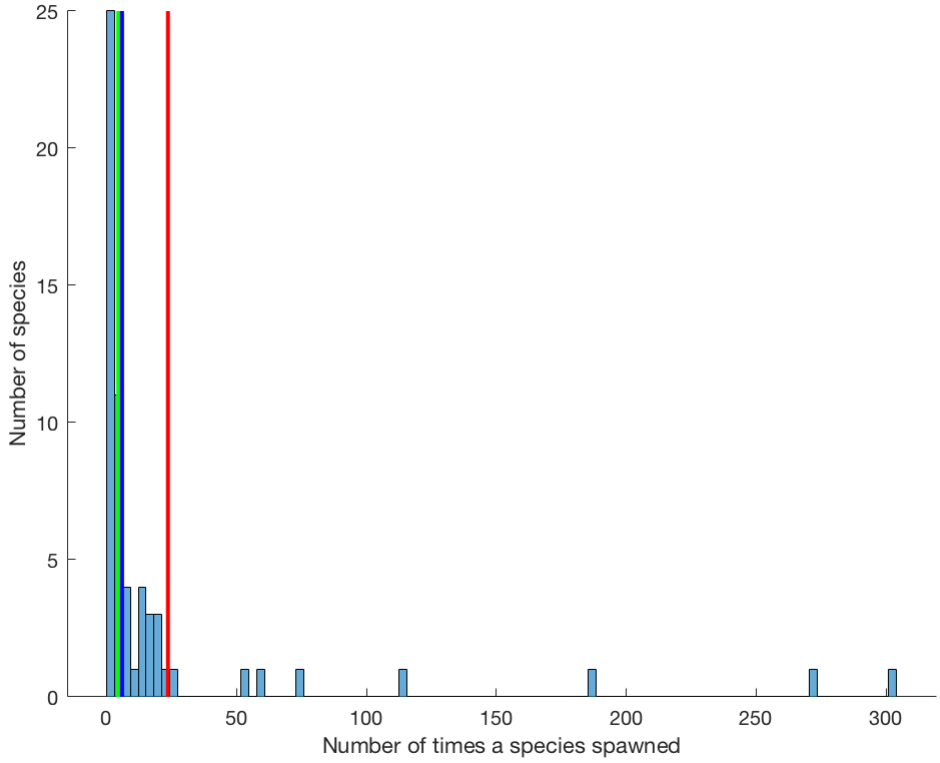
```
% INSTRUCTOR NOTE: MATLAB uses log for natural logarithms,
% but if a distribution is lognormal, it will be normally
% distributed after the logarithm with any base. Make sure to
% use the correct base when calculated the un-transformed mean value.
```

4.1.3 Calculate the un-transformed mean value and display its value on the histogram in a blue line.

```
meanOfTransformed = nanmean(log(nSpawnsObservedNans(:,iiParticipant)));
meanUntransformed = exp(meanOfTransformed);
disp(meanUntransformed);
```

```
    5.9336
```

```
plot([meanUntransformed,meanUntransformed],ylim,'b','LineWidth',2)
```



**** mean, untransformed from log transform: 5.9336 ****

4.2 Test the shape of other participants

Each participant can be thought of as running an independent experiment exploring the shape of the distribution of Pokémon spawns.

Rework your code to test each type of transformation (x^2, 1/x, log(x)) for each participant's distribution.

```
% loop over each player and test the three different
% transformation strategies
for iiParticipant = 1:nParticipants
    isNormal(iiParticipant) = lillietest(...
        (nSpawnsObservedNans(:,iiParticipant)),alphaVal);

    isSqrNormal(iiParticipant) = lillietest(...
        (nSpawnsObservedNans(:,iiParticipant)).^2,alphaVal);

    isInvNormal(iiParticipant) = lillietest(...
        1./(nSpawnsObservedNans(:,iiParticipant)),alphaVal);

    isLogNormal(iiParticipant) = lillietest(...
        log(nSpawnsObservedNans(:,iiParticipant)),alphaVal);
end
```

4.2.1 Does the same transformation that worked for 4.1.2 work for all of the other participants?

```
% check if all the values of isLogNormal are zero, meaning we fail
% to reject the null hypothesis
isAllLogNormal = all(isLogNormal==0);
disp(isAllLogNormal)
```

```
    0
```

```
% INSTRUCTOR NOTE: This is sort of a trick question, although some
% participants distributions are rejected as not lognormal, we do
% not include a proper discussion of multiple hypothesis testing
% here. In a more advanced discussion, one could include Bonferroni's
% correction for multiple hypothesis testing. Instead, we discuss it
% briefly in the next section as the transformation that has normally
% distributed data most frequently.
```

**** log transformation does not work for all participants. ****

4.2.2 Is there a transformation that works for the majority of the participants?

```
% check to see if the fraction of 0's is greater than a half, then
% most participants distributions are normal following that transformation
fracNormal = (nnz(isNormal==0))./numel(isNormal);
disp(fracNormal)
```

```
     0
```

```
fracSqrNormal = (nnz(isSqrNormal==0))./numel(isSqrNormal);
disp(fracSqrNormal)
```

```
     0
```

```
fracInvNormal = (nnz(isInvNormal==0))./numel(isInvNormal);
disp(fracInvNormal)
```

```
     0
```

```
fracLogNormal = (nnz(isLogNormal==0))./numel(isLogNormal);
disp(fracLogNormal)
```

```
    0.8370
```

**** yes, the log transformed data fail to reject the null hypothesis in almost 85% of the samples. None of the samples fail to reject the null hypothesis for any of the other transformations. ****

4.3 Comparing shapes of distributions with varying number of samples

Not every participant measured the same total number of spawns. As an extreme example, let us go back to the data from participant 60 and look at the data organized by fraction of events instead of counts of events. In this case, the *total* number of events is *one,* and the magnitude of all the measurements has been scaled down.

4.3.1 Do you get a different P-value for the lilliefors tests when you transform these data, or is the same as when you used counts?

```
% INSTRUCTOR NOTE: This question requires the students to realize
% that they have not yet made a version of the percentages without
% zero values. This question can be reworded to provide this hint
% for a less 'tricky' question.
% INSTRUCTOR NOTE: This question may require the students to realize
% that they reused the temporary index 'iiParticipant' and need to
% reset it to 60. This question can be reworded to provide this hint
% for a less 'tricky' question.

% Zeroes removed version of the frequency
percentageSpawnsObservedNan = percentageSpawnsObserved;
percentageSpawnsObservedNan(percentageSpawnsObservedNan==0) = nan;

iiParticipant = 60;
[H1,p1] = lillietest(log(nSpawnsObservedNans(:,iiParticipant)),alphaVal);
[H2,p2] = lillietest(...
    log(percentageSpawnsObservedNan(:,iiParticipant)),alphaVal);
disp(['logNormal pValue spawn counts: ',num2str(p1)]);
```

```
logNormal pValue spawn counts: 0.035133
```

```
disp(['logNormal pValue spawn frequency: ',num2str(p2)]);
```

```
logNormal pValue spawn frequency: 0.035133
```

**** Scaling from counts to frequency does not change the pValue. The same would apply to taking a logarithm with a different base, the values of the log transformed data will be different, but the shape of the distribution will be the same. ****

4.4 Utilizing transformed data

We saw earlier that the *mean* and *standard deviation* were quite sensitive to extreme values.

4.4.1 Compare the values of the *mean* on the log transformed data with and without Pidgeys. Remember to undo the transformation after calculating the *mean*.

```
meanOfTransformedNoPidgey = nanmean(...
    log(nSpawnsObservedNans(isNotPidgey,iiParticipant)));
meanUntransformedNoPidgey = exp(meanOfTransformedNoPidgey);
disp([num2str(meanUntransformed),' vs ',num2str(meanUntransformedNoPidgey)]);
```

```
5.9336 vs 5.5507
```

**** 5.9336 with Pidgeys, 5.5507 without ****

For the standard deviation, calculate the value of the exp(mean + standard deviation) - exp(mean). This moves to one standard deviation above the mean before undoing the transformation, but you still want to subtract off the mean to give a value that only has to do with the spread of the data.

4.4.2 What's the equivalent to one standard deviation of the log transformed data with and without Pidgeys?

```
stdOfTransformed = nanstd(log(nSpawnsObservedNans(:,iiParticipant)));
stdOfTransformedNoPidgey = nanstd(...
    log(nSpawnsObservedNans(isNotPidgey,iiParticipant)));
stdUntransformed = exp(meanOfTransformed+stdOfTransformed)-...
    exp(meanOfTransformed);
stdUntransformedNoPidgey = exp(...
    meanOfTransformedNoPidgey+stdOfTransformedNoPidgey)-...
    exp(meanOfTransformedNoPidgey);
disp([num2str(stdUntransformed),' vs ',num2str(stdUntransformedNoPidgey)]);
```

```
20.8324 vs 17.5797
```

**** 20.8324 with Pidgeys, 17.5797 without ****

4.4.2 Does working with the properly transformed data help provide some stability against the extreme values?

****Yes, because the standard deviation is a good measure of spread for describing a normal distribution. ****

Measuring biodiversity

There are multiple ways to characterize the biodiversity in a specific area. These metrics apply to any method of sampling species in a fixed area, not only Pokémon GO.

5.1 Shannon index

Side note: Somewhat confusingly for this problem set, the variable *H* is used as both the name of the Shannon index, and the boolean variable *H* from section 4 where it was the result of a hypothesis test. These are two different *H*'s and are not related to each other.

The Shannon index measures the uncertainty in predicting the identity of a sample chosen at random from the population. As the population becomes more diverse, the Shannon index goes up, as do the uncertainty as to what the identity of the random sample will be. The Shannon index is calculated as the sum $H = -\Sigma(\mathrm{p} \cdot \log(\mathrm{p}))$ where *p* is the frequency of events and the sum is performed over all species. Calculate the Shannon index for each of the players' collection.

```
% this can be done as a loop over each participant, or in a single
% line as an elementwise matrix manipulation
ShannonIndex = -1*nansum(...
    percentageSpawnsObservedNan.*log(percentageSpawnsObservedNan));

% INSTRUCTOR NOTE: Because we have removed all of the zero values
% before calculating this, the Shannon index is fairly numerically
% stable. In some situtations, when p is very small, (p log p) can
% be unstable, and so a lower threshold can be used to set p log p to 0.
```

5.1.1 Which participant has the most diverse collection? What is the value of their Shannon index?

```
[maxShannonIndex,mostDiverseByShannon] = max(ShannonIndex);
disp(mostDiverseByShannon);
```

```
    17
```

```
disp(maxShannonIndex);
```

```
    3.5972
```

**** participant 17, Shannon Index = 3.5972 ****

5.1.2 Use the participants self-reported location to determine the human readable name of this location with the greatest Shannon index.

```
disp(locationList{mostDiverseByShannon});
```

```
  Coffs Harbour
```

**** Coffs Harbour ****

5.2 Simpson index

The Simpson index is a measurement of how likely it is that two random samples from the distribution will have the same identity and is calculated as $\lambda = \Sigma(p^2)$. A distribution that has no diversity will have a single non-zero *p* (*p*=1) and therefore a Simpson index of 1. A very diverse population will have many values of *p*<1 and therefore a Simpson index close to 0.

5.2.1 Which participant has the most diverse collection? What is the value of their Simpson index?

```
SimpsonIndex = nansum(percentageSpawnsObservedNan.^2);
[maxSimpsonIndex,mostDiverseBySimpson] = min(SimpsonIndex);
[minSimpsonIndex,leastDiverseBySimpson] = max(SimpsonIndex);
disp(mostDiverseBySimpson);
```

```
    61
```

```
disp(maxSimpsonIndex);
```

```
    0.0380
```

**** Participant 61, Simpson Index of 0.038 ****

5.2.2 What are the latitude/longitude of the location with the **least** diverse collection?

```
disp(['Latitude: ',num2str(latitudeList(leastDiverseBySimpson)),...
```

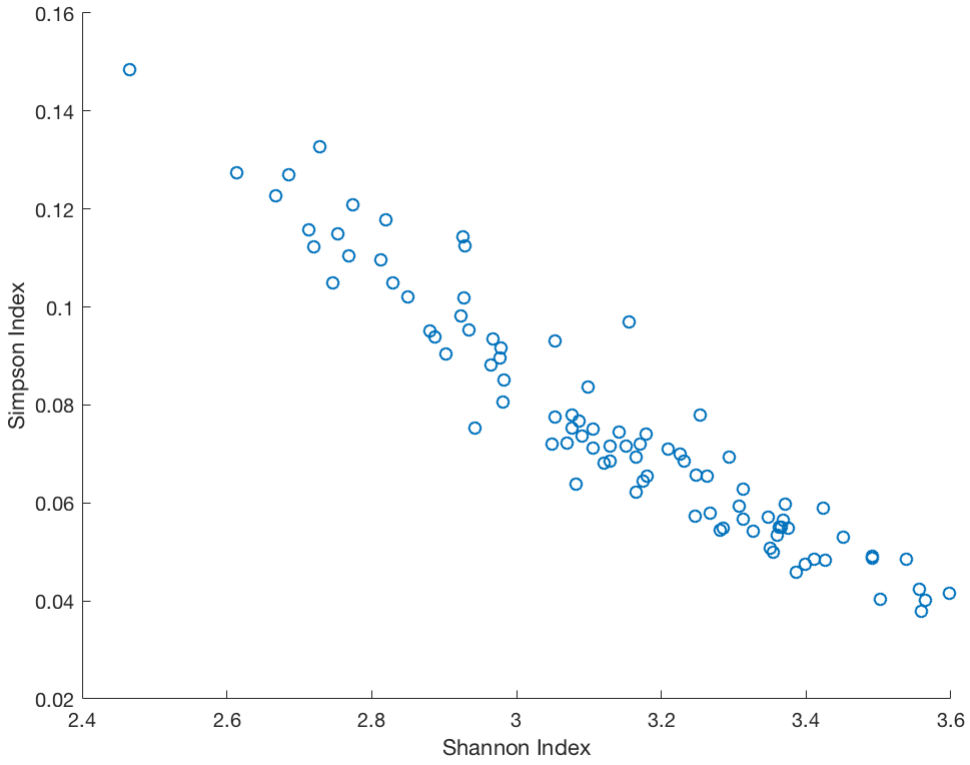
```
Latitude: 51.2277. Longitude: 6.7735
```

```
    '. Longitude: ',num2str(longitudeList(leastDiverseBySimpson))]);
```

**** Latitude: 51.2277, Longitude: 6.7735 ****

5.2.3 Make a plot of the Simpson Index vs. the Shannon Index and describe the relationship between the two.

```
figure(gcf);
clf;
scatter(ShannonIndex,SimpsonIndex);
xlabel('Shannon Index');
ylabel('Simpson Index');
```



**** There is a strong negative correlation between the two measures of diversity. This makes sense because larger Shannon indices and smaller Simpson indices both indicate samples with greater diversity. ****