Group Project: Bigfoot
Due: April 20th

Goals

1. Demonstrate mastery of data science tools (python, data frames, visualizations)
2. Communicate results to a general audience

Description

Your team will use the tools learned in MA705 to conduct and present a descriptive statistics analysis of Bigfoot records. Bigfoot, also known as Sasquatch, is an ape-like creature purported to inhabit the forests of North America. Supposed evidence of Bigfoot, including sightings, photographs, footprints, etc, have been presented throughout the 20th century.

While the credibility of such evidence may certainly be questioned, the influence of the Bigfoot on US culture and perhaps reflection of it cannot be. What does an analysis of the Bigfoot sightings data set reveal? Your team's analysis will be presented in a Jupyter notebook and include several visualizations.

Motivating Questions

- What social/economic/geographic/climate/political characteristics of a US county are associated with Bigfoot sightings?
- What trends are there in the number of reported Bigfoot sightings over time?
- Is there a common narrative or a common physical description of Bigfoot?

Key Steps

- Understand the data: The primary data set, assembled by the Bigfoot Field Researchers Organization (BFRO)[1], contains 4747 records of Bigfoot sightings in the US. Variables include text of the report, latitude/longitude of the sighting, sighting date, number (a report id), classification (see BFRO link), and a variety of weather conditions (added later for some reports based on weather records).
- Consider bringing in additional data. Detailed US county geospatial data is being provided, as well as some county level summary statistics. **Your team is encouraged to incorporate other county-level data sets that you believe would be interesting**.

[1] http://www.bfro.net/GDB/classify.asp

- Conduct a thorough descriptive statistics analysis of the data using the tools of MA705. At a minimum, in preparation you'll need to: decide how to deal with missing data; create a geospatial data frame from the Bigfoot reports; identify which US county each sighting took place in; eliminate sightings outside the contiguous US states; and prepare a county data set with whatever county-level variables you intend to work with. To be clear, there are likely two data frames you're working with: One with Bigfoot reports (made geospatial and perhaps with added variables) and one with US counties as rows (also geospatial, with polygon outlines for the counties, and with added variables that might include Bigfoot sighting counts as well as other variables for comparison).
- Investigate the data. Compare county-level variables with Bigfoot sightings and decide on one or a small number of comparisons to focus on. Separately, investigate the text of the reports. Create a word cloud either for the full text of the reports, quite possibly using a set of "stop words" of your choosing. Also extract (write to .txt files?) phrases that describe Bigfoots height or weight and summarize these either by inspecting the list, with a word cloud, or some visualization if you think it appropriate.
- Write a Jupyter Notebook explaining your project to an outsider with only general knowledge. The notebook should be professional with the visualizations incorporated. **It is recommended you do the entire project in Google CoLab, for collaboration purposes and to make sure there are no compatibility issues.**

Deliverables

- **Primary:** A well-written Jupyter notebook containing an introduction to the project and data, a brief discussion of the methodology for your analysis accessible to a general audience, a presentation of the results, and a brief conclusion. It does not need to be especially long, but it should be clear and the visualizations should be excellent.

  You'll need the following visualizations incorporated into the notebook:

  - A map of the contiguous US illustrating the **locations of Bigfoot sightings**.
  - A map of the contiguous US illustrating the **Bigfoot sightings per county**.
  - Two appropriate visualizations that investigate the number or rate of Bigfoot sightings per county to other county-level social/economic/geographic/climate/political variables that you find interesting. There is no specific type of visualization required: It might be a map, bar plot, scatter plot (with best fit curve of some type?), etc, and it might be based on statistical modeling learned in other courses (although this is not a requirement and not doing so will not hurt your grade).
  - A visualization of Bigfoot sightings per year (a time series).
  - A word cloud based on appropriately filtered text that gives a sense of common narrative or description of Bigfoot. How you focus your word cloud is up to you.

  Excellent data visualization will count heavily toward final grade.

- **Other:** Supporting `.py` files for preparing data and generating visualizations, any external data sets, and a list of references. In short, I should be able to check your team's work.

Guidelines

- Focus only on the contiguous United States (i.e. excluding Hawaii and Alaska). Do not include US and Hawaii. Do not include Canada. Do not include water sightings (yes, there are some).
- The report must be handed in as an exported Jupyter notebook. You should export the notebook as a html while hiding the code and prompts. This can be done locally:

```
jupyter nbconvert filename.ipynb --no-input --no-prompt --to html
```

- The project should be mathematically correct, professional, engaging, efficient, and written at the correct level. **Correct level means that a reader with only a general education should be able to follow it**. I'm not interested in how much statistics or python you can show off, and in fact doing so will hurt your grade. In your professional life, you'll need to get the statistics and programming right on your own and communicate the results to people who aren't professors!
- On that note, undigested computer output is definitely not appropriate. There will not be any code in your notebook, since the code is hidden.
- When discussing methodology, focus on what would interest your reader or be important to them. How did you make your choices, not how did you write your code.
- Any plots should be fully labeled and also referenced in the report.

Further Comments

- Selected background:
  - http://www.bfro.net/GDB/classify.asp
  - https://www.smithsonianmag.com/history/why-so-many-people-still-believe-in-big
- You **must** use python to prepare the data and Jupyter notebooks to write the report.
- Your team should almost exclusively use python tools that were covered in MA705. Anything that significantly alters the spirit of the project will be negatively graded.

Questions and Answers

How do we handle missing data?
It's up to your group. The first step should **not** be to delete anything with missing data. The parts of the project are fairly independent of each other, so a Bigfoot observation that is missing, say, location information would still be useful for its description or date. Your client would probably need to know about missing data, how you dealt with it, and how you feel it affected the analysis (if at all in terms of bias).

How do we open the counties geospatial data?
The `county_geo.zip` file contains detailed geospatial data at the county level for the US. Unzip it to a folder, then put that folder in the same directory as your python file (or on CoLab). The only file you need to directly use is the .shp file, but the others are necessary. Read it in the same way as the usa state data in Homework 5.

What are FP/FIPS?
FP (also called FIPS) are unique codes used to identify US geographical units. State level and territory level ones are here: here and county level ones are here. The variable GEOID in the counties data set is the unique five digit county code.

How do we read/write a pickle file?
Exactly like .csv files. There is a `pd.read_pickle` and a `df.to_pickle` function. The advantage of pickle is that it saves something that originated in python with all the data types, so when you read it again you don't need to reformat. For instance, if a FIPS was the string "01234" and you saved it to .csv, it would then get read in as the number 1234. But with pickle format it would be read back in as the string "01234". Even geospatial data sets get read back in as geospatial data sets. After you clean your bigfoot data and county data, I recommend saving them in pickle format and then loading into new programs to do each of the visualizations/analyses.

Warning: CoLab can only read "protocol 4", which is not the default way pickle files are saved. I updated the `county_label.pkl` file to be saved in protocol 4. If you want to save your own pickle file in CoLab so that CoLab can read it in the future, use `df.to_pickle(FILENAME, protocol = 4)`.

How do we line up geospatial county data with other county data?
There is a `county_labels.pkl` file that contains a data frame with county FIPS, county name, state abbreviation, and full state name. Be aware that the word "County" is never part of the county name in this file, whereas it may or may not be elsewhere. This means that some counties might be called "New York County" elsewhere, but officially just "New York" (the county) in `county_labels.pkl`.

How do we restrict US counties to contiguous US states?
The county geospatial data has a STATFP column. You can quickly make a list of the non-contiguous ones (Alaska, Hawaii, and US Territories), or with slightly more typing make a list of the contiguous ones. Remove rows for non-contiguous STATEFP codes permanently from the counties geospatial data frame. The map will now plot nicely as 48 contiguous states with county outlines.

How do we restrict Bigfoot sighting to contiguous US states?

Start by making a geopandas data frame for only the bigfoot rows that have latitude/longitude. Make a single polygon for the contiguous US. The best way to make a union of polygons is:

```
from shapely.ops import unary_union
usa = geopandas.GeoSeries(unary_union(counties.geometry))
```

Now use the `.within()` method to generate a True/False list for whether each sighting is in the contiguous US polygon (as in the Week 6 Prepare Airplane Crash Data, except you don't need to do a lot of the for loop stuff). The syntax is `POINT.within(POLY)` to decide True/False if a point is within a polygon. Here the POLY will be the usa just defined. Use this True/False list to permanently re-define the bigfoot data set to be just contiguous US sightings. You'll find there are 3768 bigfoot sightings that take place in the contiguous US, i.e. your plot will have 3768 bigfoot sightings.

How do we decide which county a bigfoot sighting takes place in??!?!
You need to decide which county polygon each sighting's point is in, and then record that county's FIPS in a new variable. More specifically, run a for loop over the bigfoot geospatial data. For each sighting, check if the latitude/longitude point is contained in each county polygon (it will only be in one). The for loop in Week 6 Prepare Airplane Crash Data is very relevant, although you don't need the if/else part. Basically, sightings are crashes and counties are countries. Write that county's FP/FIPS identifier to a new variable in the bigfoot data set. THIS TAKES SEVERAL MINUTES TO RUN (just think about how many points the computer is checking are in so many polygons!). Perhaps test it on a small set of the Bigfoot sightings until you know it's working correctly. If you run a counter on the resulting variable, you'll find that county 53053 (Pierce County in WA) has the most bigfoot sightings (62).

The data preparation takes a long time to run on the computer. Can we make it faster?
It's doing a lot of searching. Write one program just to prepare data. Let it run, then save the cleaned data frame as a file that you read in your other parts of the project. The best way to do this is `DATAFRAME.to_pickle('FILENAME.pkl')`. Saving as data frame (or geodataframe) in pickle format will preserve the geospatial data as well as the type of columns (such as strings, so that when you load again the COUNTYFP leading zeros don't disappear). Put simply, at the end of your program that cleans the data, you'd have

```
bigfoot_geo.to_pickle('bigfoot_geo.pkl')
counties.to_pickle('counties_clean.pkl')
```

and at the top of your other programs you'd have

```
counties = pd.read_pickle('counties_clean.pkl')
bigfoot_geo = pd.read_pickle('bigfoot_geo.pkl')
```

How do we make a word cloud?
It's easy! Install the library wordcloud. Then use the syntax

```
wordcloud = wordcloud.WordCloud(max_words = 20, background_color='green').generate(TEXT
```

to make a wordcloud with, say, 20 words out of the text contained in TEXT. You then use

```
plt.imshow(wordcloud, interpolation="bilinear")
```

to plot the wordcloud, and of course you can use all your usual matplotlib tools. The finer detail here is preparing the TEXT well and perhaps using the stopwords parameter that allows you to omit a list of words that you know are unrelated.
Wordcloud example:

```
import wordcloud
import matplotlib.pyplot as plt

x = "test hi one test two test three hi"
y = ["hi", "one"]

cloud = wordcloud.WordCloud(max_words = 2, stopwords=y).generate(x)
plt.imshow(cloud, interpolation="bilinear")
```

Specifying a list of stopwords isn't necessary, but it makes a more relevant word cloud.

Is it possible to save sessions on CoLab?
You can save the notebook as a .ipynb to your Google Drive (or elsewhere), but it's not possible to maintain and active virtual session. Any supporting files you add will be deleted. The best way to manage things is keep a folder on your desktop with all relevant files. Drag/drop all of them together to CoLab's folder when you start a session.

We're using CoLab. How do we export our final .ipynb as .html with the python code hidden?
Get your final .ipynb as you want it to look. Run all the cells in CoLab and save a copy of the .ipynb to desktop. Start a new notebook in CoLab. Drag the .ipynb to the files folder for that notebook, so the new notebook can see it. In a python cell of the new notebook, type and run

```
!jupyter nbconvert BIGFOOTPROJECT.ipynb --no-input --no-prompt --to html
```

A .html will appear in the files folder. Save it to your computer and check that it looks correct.

More Help: Processing years in the Bigfoot project

- There is one date recorded as the string `'1869-11-10'`. It's from an old newspaper article (if you check the description). For the purposes of plotting sightings per year nicely, drop this sighting. It's such an outlier that it takes away from the visualization.
- All other years are supposed to be between 1921 and 2018.
- However, they're currently strings in the format MM/DD/YY.
- When python guesses a four-digit year based on two-digit year when converting to datetime, it chooses 19YY if YY is 69 or greater and 20YY otherwise. This isn't always right for our data.

```
datetime.datetime.strptime('68', '%y')
Out[1]: datetime.datetime(2068, 1, 1, 0, 0)

datetime.datetime.strptime('69', '%y')
Out[2]: datetime.datetime(1969, 1, 1, 0, 0)
```

- So you'll need to manually set a four-digit year. Luckily, you don't need to do surgery on the date variable since to plot sightings per year all you need is a new column in the bigfoot data frame that has the right four-digit year. Once you have it, you can `bigfoot.groupby('year').count()` and plot away.
  - Load the cleaned bigfoot data into your sightings per year file. (Separate files for each part of the project is recommended)
  - Drop the 1869 row
  - Drop all rows with no date
  - Define a new variable `year` by considering the two-digit YY appropriately as above.

  Here is what part of the plot would look like: