Assignment 4

Goals

The goal of this assignment is to use pandas to perform some data cleaning tasks.

Instructions

In this assignment, we will clean the raw HURDAT2 dataset for Atlantic hurricane data from 1851 through 2017. This dataset is provided by the National Hurricane Center and is documented here. Our goal is to get it into a format like the hurdat2.csv file used for Assignment 2 and Assignment 3.

Due Date

The assignment is due at 11:59pm on **November 16**

Submission

You should submit the completed notebook file required for this assignment on Canvas. The filename of the notebook should be `a4.ipynb`.

Details

1. Reading Data and Naming Columns (10 pts)

Read the data from `hurdat2.txt` using panda's `read_csv` method and name the columns accordingly. If you do this assignment in the same directory as Assignment 1, you should still have the `hurdat2.txt` file locally. If not, you can run the following code to download it:

```
from urllib.request import urlretrieve
url = "https://www.nhc.noaa.gov/data/hurdat/hurdat2-1851-2017-050118.txt"
local_fname = "hurdat2.txt"
if not os.path.exists("hurdat2.txt"):
    urlretrieve(url, local_fname)
```

If you try to read the data using just the filename, you will get a strange result where each row has a really long label. Remember that the format looks like:

```
AL011851,            UNNAMED,     14,
18510625, 0000,  , HU, 28.0N,  94.8W,  80, -999, -999, -999, -999, -999, -999, -999, -999, -999, -999, -999, -999, -999,
18510625, 0600,  , HU, 28.0N,  95.4W,  80, -999, -999, -999, -999, -999, -999, -999, -999, -999, -999, -999, -999, -999,
```

alternating between hurricane identification information and specific tracking points. So (a) there is no header specifying what the columns store, and (b) there are different numbers of fields in each row depending on whether the row is hurricane identifier information or tracking points. Read in twenty columns and name them as follows:

```
['date','time','record_id','status','latitude','longitude','max_wind','min_pressure','ne34ktr','se34ktr','sw34ktr','nw34ktr','ne50ktr','se50ktr','sw50ktr','nw50
```

By default, pandas will parse rows like `"    UNNAMED, HU"` and **keep** the leading whitespace so you get values like `"    UNNAMED"` and `" HU"`. We **do not** want this extra whitespace. Luckily, there is a `read_csv` option to remove it (find it in the docs). The output should look like:

	date	time	record_id	status	latitude	longitude	max_wind	min_pressure	ne34ktr	se34ktr	sw34ktr	nw34ktr	ne50ktr	se50ktr	sw
0	AL011851	UNNAMED	14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	18510625	0000		HU	28.0N	94.8W	80.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-9
2	18510625	0600		HU	28.0N	95.4W	80.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-9
3	18510625	1200		HU	28.0N	96.0W	80.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-9
4	18510625	1800		HU	28.1N	96.5W	80.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-9

Hints:

- Check that the DataFrame looks correct by displaying it. If the variable `df` references the DataFrame, just typing `df` in a cell will display the table.
- Remember to tell pandas **not** to read in a header for this CSV file (see the `read_csv` docs)
- If you don't provides the list of column names (see the `read_csv` docs), `read_csv` will complain about having too many columns. If you specify that the data has the twenty columns listed above, it will fill in empty columns with `NaN` values.

2. Extract and Fill Identifiers (15 pts)

Clearly, we still have an issue in that the rows that are hurricane identifiers are mixed in with rows that have tracking information. We want this information to attached to each tracking point row. This involves four steps:

1. Extract out those rows with the identifier, name, and number of points, and put them in new columns named named `identifier`, `name`, and `num_pts`
2. Fill in this information for the tracking points
3. Remove the rows that just contain identifier information
4. Move the new columns to the front of the data frame.

The final data should look like:

	identifier	name	num_pts	date	time	record_id	status	latitude	longitude	max_wind	min_pressure	ne34ktr	se34ktr	sw34ktr	nw
1	AL011851	UNNAMED	14	18510625	0000	NaN	HU	28.0N	94.8W	80.0	-999.0	-999.0	-999.0	-999.0	-
2	AL011851	UNNAMED	14	18510625	0600	NaN	HU	28.0N	95.4W	80.0	-999.0	-999.0	-999.0	-999.0	-
3	AL011851	UNNAMED	14	18510625	1200	NaN	HU	28.0N	96.0W	80.0	-999.0	-999.0	-999.0	-999.0	-
4	AL011851	UNNAMED	14	18510625	1800	NaN	HU	28.1N	96.5W	80.0	-999.0	-999.0	-999.0	-999.0	-
5	AL011851	UNNAMED	14	18510625	2100	L	HU	28.2N	96.8W	80.0	-999.0	-999.0	-999.0	-999.0	-

Hints:

- To select identifier rows, look at the first few characters of the `date` field, and use a boolean expression on the **string** representation (`.str`) of that field.
- Remember that new columns can be copied/created using bracket assignment (`df['newcol_name'] = ...`). Multiple columns can be assigned using a list of columns on the left-hand side.
- Use `fillna` on only the new columns that need to be filled and use the proper fill method (see the `fillna` docs)
- To select those rows that are **not** identifier rows, you can **negate** (`~`) the boolean expression used to select them.
- To reorder columns, you can just pass the new order to the dataframe (`df[['c4','c5','c1','c2','c3']]`). Use slicing on `df.columns` to do this without rewriting all the column names.

3. Cleanup (30 pts)

There are four cleanup tasks: (a) replace `"UNNAMED"` with `NaN`, (b) replace `-999` with `NaN`, (c) convert the date and time to a **timestamp**, and (d) change `latitude` and `longitude` to numeric representations.

a. Replace `UNNAMED` and `-999` (10 pts)

We wish to replace hurricane names of `UNNAMED` with `NaN`, and any of the wind speed radii measures of `-999` with `NaN`.

Hints:

- The `replace` method will work over the entire data frame or over a particular column.
- Pandas read in the -999 value as a float so you may need to construct the value to be replaced as a float.
- Remember to either update the DataFrame in place or update the reference to the updated DataFrame.

b. Create a timestamp (10 pts)

Right now, we have two columns for date and time. This makes it difficult to calculate the amount of time between two different hurricane tracking points. If we convert them to a timestamp, such calculations are easy. To do this, we can use pandas' `to_datetime` method. This method can convert from strings to timestamps. In our case, if we concatenate the date and time columns, and feed the concatenated series to `to_datetime`, things should work. Add the new column named as `datetime` and remove the old `date` and `time` columns.

Hints:

- To concatenate strings, you can have pandas treat the column as a string using `.str[:]`. Then you can use standard string operators like `+` for concatenation.

c. Convert Latitude and Longitude (10 pts)

In the original data, latitudes are specified like `28.0N` or `34.5S` while longitudes are `94.8W` or `4.5E`. Most systems store such values as negative or positive numbers where `S` latitudes and `W` longitudes are stored with a negative sign. We will use pandas methods to convert these values to that format. In this case, we want to end up with a float type for these columns at the end. There are different ways to do the conversion, but one method is to use string indexing (`.str[<index-or-slice>]`) to extract out the last character (N,E,S,W), convert that to a boolean (use a comparison), and then convert the boolean to a number via `.astype(int)`. True values become 1 and false values become 0, but we will need 1 and -1. Multiply the rest of the value (converted to a float) by the indicator (-1 or 1) to obtain the desired result.

Hints:

- You can use `astype` to convert a string column to floats
- If you have a value in [0,1] and want to scale it to [-1,1], multiply by 2 and then subtract 1.
- Remember to put the computed columns back into the data frame, replacing the original columns.

After all of these changes, your data frame should look something like:

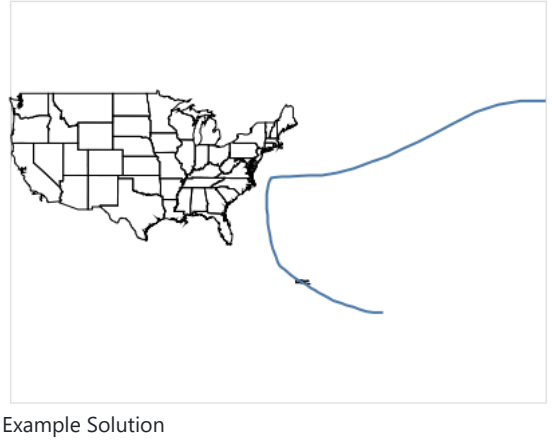
	identifier	name	num_points	datetime	record_id	status	latitude	longitude	max_wind	min_pressure	ne34ktr	se34ktr	sw34ktr	nw34ktr
1	AL011851	NaN	14	1851-06-25 00:00:00	NaN	HU	28.0	-94.8	80.0	NaN	NaN	NaN	NaN	NaN
2	AL011851	NaN	14	1851-06-25 06:00:00	NaN	HU	28.0	-95.4	80.0	NaN	NaN	NaN	NaN	NaN
3	AL011851	NaN	14	1851-06-25 12:00:00	NaN	HU	28.0	-96.0	80.0	NaN	NaN	NaN	NaN	NaN
4	AL011851	NaN	14	1851-06-25 18:00:00	NaN	HU	28.1	-96.5	80.0	NaN	NaN	NaN	NaN	NaN
5	AL011851	NaN	14	1851-06-25 21:00:00	L	HU	28.2	-96.8	80.0	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
52146	AL192017	RINA	21	2017-11-08 12:00:00	NaN	TS	38.3	-48.8	45.0	994.0	160.0	160.0	0.0	90.0
52147	AL192017	RINA	21	2017-11-08 18:00:00	NaN	TS	40.1	-49.0	45.0	992.0	180.0	180.0	0.0	100.0
52148	AL192017	RINA	21	2017-11-09 00:00:00	NaN	TS	41.8	-48.8	45.0	991.0	180.0	200.0	0.0	100.0
52149	AL192017	RINA	21	2017-11-09 06:00:00	NaN	LO	43.6	-48.0	40.0	993.0	160.0	210.0	0.0	120.0
52150	AL192017	RINA	21	2017-11-09 12:00:00	NaN	LO	45.5	-47.0	40.0	995.0	160.0	240.0	0.0	140.0

4. Use the Data (15 pts)

Write the new data out to a CSV file and create an altair visualization that shows the path of Hurricane Maria in 2017. You **must** create a CSV file to receive full credit. **Do not** use the data frame directly. Show the US states and Puerto Rico (as in Assignment 2) to provide context and see how Maria's path took it directly across the island.

Hints

- Use the `to_csv` method to write out the data.
- When you write the data, you may need to use the `date_format` parameter to write dates out with a "T" in between the date and the time. For example, `1851-06-25T06:00:00`. Refer to http://strftime.org to determine how to create the format string. Note that you can add characters like "T". "-", and ":" as the literal characters in such strings.
- Refer back to Assignment 2 for code to draw the US states and Puerto Rico.
- You need to filter by both hurricane name and year to only show Maria 2017.



Example Solution