Assignment 5

Goals

The goal of this assignment is to work with time series data and perform sampling and aggregation tasks with pandas.

Instructions

In this assignment, we will analyze a cleaned version of the **HURDAT2 dataset** for Atlantic hurricane data from 1851 through 2017. This dataset is provided by the National Hurricane Center and is documented here. We will use the cleaned data from **Assignment 4** for this assignment without the `ktr` fields.

Submission

You should submit the completed notebook file required for this assignment on myCourses. The filename of the notebook should be firstname_lastname_a5.ipynb.

Due Date

The assignment is due at 11:59pm on **December 3**

Details

In this assignment, we will use the hurricane data, but in a cleaned up CSV format. Note that we will use the updated version like the one you created in A4 (**not** A2). You can download the **`hurdat2.csv`** data from myCourses. Each data item is a specific point along the hurricane's trajectory, and unlike in the original data, **every** data item contains the hurricane's identifier and name. The beginning of this file looks like:

	identifier	name	num_pts	datetime	record_id	status	latitude	longitude	max_wind	min_pressure
0	AL011851	NaN	14	1851-06-25 00:00:00	NaN	HU	28.0	-94.8	80.0	NaN
1	AL011851	NaN	14	1851-06-25 06:00:00	NaN	HU	28.0	-95.4	80.0	NaN
2	AL011851	NaN	14	1851-06-25 12:00:00	NaN	HU	28.0	-96.0	80.0	NaN
3	AL011851	NaN	14	1851-06-25 18:00:00	NaN	HU	28.1	-96.5	80.0	NaN
4	AL011851	NaN	14	1851-06-25 21:00:00	L	HU	28.2	-96.8	80.0	NaN

and the fields are:

- `identifier` is a unique identifier for each hurricane
- `name` is the hurricane's name
- `num_pts` is the number of points recorded for the hurricane
- `datetime` is the date and time of the recorded point (in Coordinated Universal Time (UTC))
- `record_id` is the record identifier as defined by the documentation
- `status` is the status of the system as defined by the documentation
- `latitude` is the latitude of the recorded point
- `longitude` is the longitude of the recorded point
- `max_wind` is the maximum sustained wind (in knots)
- `min_pressure` is the minimum pressure (in millibars)

1. Aggregation (20 pts)

a. Windspeed and Pressure (10 pts)

Aggregate the data so that each hurricane is shown with its maximum windspeed and minimum pressure over all recorded points.

Hints

- Make sure you use different aggregations for windspeed and pressure, and think about which aggregation makes the most sense.
- The form of `.agg` that takes a dictionary may be useful.

b. Hour counts (10 pts)

Glancing at the data, we might think that the data is reported every six hours at midnight, 6am, 12pm, and 6pm. Check to see if this is true by calculating which hours appear in the timestamps. Which hour occurs least often?

Hints:

- Make sure that you have parsed the datetime field as a datetime (`read_csv` has an option to convert specific columns to datetime)
- Remember to convert the timestamps to allow components of the timestamp (e.g. the hour) to be accessed.
- You can group by components without first storing them in a column
- Remember that `size` can count the number of items in each group.

2. Speed (30 pts)

Calculate the speed of the hurricane between the measured points. You will need to group the data, make sure it is sorted, and then calculate the speed at each point as distance / time. This is probably most straightforward to calculate in three steps: (a) Calculate the time difference between consecutive points; (b) Calculate the distance between consecutive points; and (c) divide the distance by time to obtain speed. It is best to store the individual values for (a) and (b) and use the columns to compute (c). Use the following haversine distance formula to calculate the distance in kilometers based on latitude and longitude (from StackOverflow):

```
import numpy as np
def haversine(lon1, lat1, lon2, lat2, earth_radius=6367):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)

    All args must be of equal length.

    """
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = np.sin(dlat/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2.0)**2

    c = 2 * np.arcsin(np.sqrt(a))
    km = earth_radius * c
    return km
```

For both time and distance, we need to make sure that the points do not come from different hurricanes. Make sure to **group** the points by hurricane before calculating distance or time differences. For time, the built-in `diff` function should be useful. For distance, you will need to use a custom `apply` that passes shifted latitude and longitude series to the second arguments for the `haversine` function. One trick is that grouping by identifier may require promoting the original identifier (row id) back to the index after the apply is complete. Finally, you should divide the distance by the time difference. The units for the speed should be kilometers per hour.

An example solution is

	identifier	name	num_pts	datetime	record_id	status	latitude	longitude	max_wind	min_pressure	time_diff	dist	speed
0	AL011851	NaN	14	1851-06-25 00:00:00	NaN	HU	28.0	-94.8	80.0	NaN	NaT	NaN	NaN
1	AL011851	NaN	14	1851-06-25 06:00:00	NaN	HU	28.0	-95.4	80.0	NaN	06:00:00	58.870532	9.811755
2	AL011851	NaN	14	1851-06-25 12:00:00	NaN	HU	28.0	-96.0	80.0	NaN	06:00:00	58.870532	9.811755
3	AL011851	NaN	14	1851-06-25 18:00:00	NaN	HU	28.1	-96.5	80.0	NaN	06:00:00	50.279389	8.379898
...	...	...	...	...	...	...	...	...	...	...	...	...	...
50299	AL192017	RINA	21	2017-11-08 18:00:00	NaN	TS	40.1	-49.0	45.0	992.0	06:00:00	200.765152	33.460859
50300	AL192017	RINA	21	2017-11-09 00:00:00	NaN	TS	41.8	-48.8	45.0	991.0	06:00:00	189.656830	31.609472
50301	AL192017	RINA	21	2017-11-09 06:00:00	NaN	LO	43.6	-48.0	40.0	993.0	06:00:00	210.421792	35.070299
50302	AL192017	RINA	21	2017-11-09 12:00:00	NaN	LO	45.5	-47.0	40.0	995.0	06:00:00	225.495431	37.582572

Hints

- Calculate the distance as a separate object first, then figure out how to fit it that distance back into the original data frame.
- The original index should be accessible in the calculated distance, but you may need to reset the index.
- The Timedelta calculated by `diff` can be obtained as different values, but the `total_seconds` method can be converted to the total number of hours (a value that does not exist).
- `idxmax` can be useful in locating the point (and thus name of the storm) with the highest speed.

3. Resampling and Visualization (50 pts)

In this problem, we will compare two hurricanes, Joaquin (2015) and Maria (2017), and visualize their speeds over time so that we can see similarities or differences. This will require shifting the timestamps and resampling so direct comparisons can be made. Generate all visualizations using altair and passing dataframes directly to altair.

a. Joaquin vs. Maria (10 pts)

Let's examine the speed of both Hurricane Maria and Hurricane Joaquin. Plot each hurricane's speed over time individually using altair as a line chart.

Hints

- Since we will be using the datasets for Joaquin and Maria throughout, it will make sense to select the data for each hurricane and store it in a separate data frame.
- Remember that a hurricane name can be **reused** in different years.

b. Superimposed Plots (10 pts)

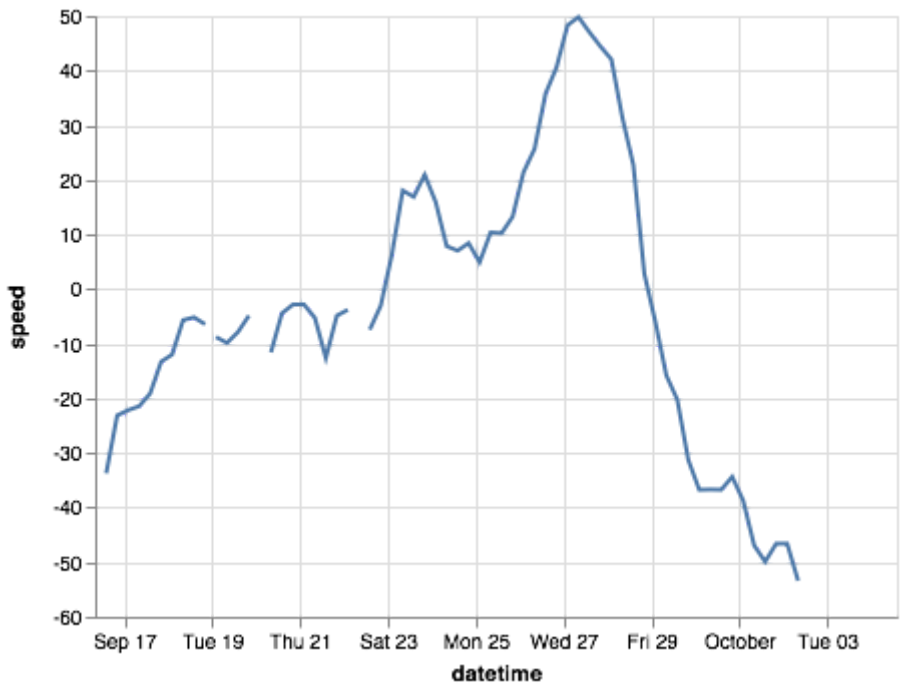
Rather than compare Joaquin and Maria side-by-side, let's overlay them. One issue is that they have different timestamps, but we can shift Joaquin's dates to line up with Maria's.

Hints:

- To shift dates, we can again use the **shift** method, but now we wish to do this for a specific amount of time (the difference between Joaquin's first datetime and Maria's first datetime).
- Altair's composition operator (+) will do superimposition.
- Make sure each line is a different color

c. Difference Plot (10 pts)

Now show the difference between the speeds in a separate plot. This will look broken in that there are gaps in the line graph. Don't do anything special to fix this for now.



Example Solution

Hints

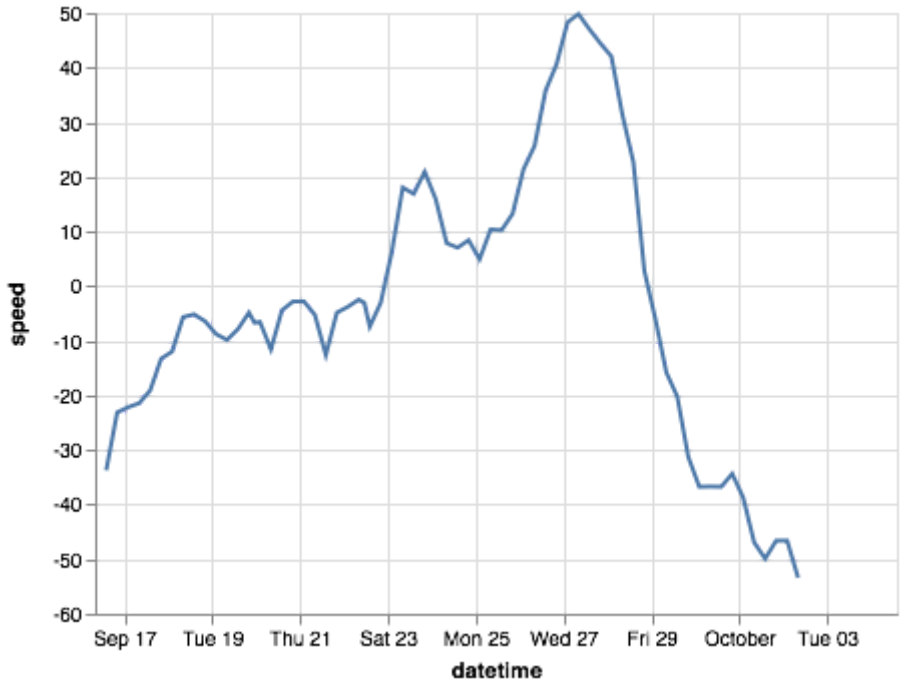
- Make sure you are using the shifted data when computing the difference

d. Resampling (20 pts)

The reason for the gaps is that Joaquin has some non-standard timestamps. We can fix this by resampling the data. Resample the speed every three hours and interpolate to fill in any gaps. Now create the same difference plot. The gaps should be gone.

Hints

- You need to both resample and then interpolate to fill in the gaps
- Do the resampling on both hurricane's data **individually**, then recompute the difference.



Example Solution