NREL Technical Task 2022

The Task:

Wind resource varies highly across space and at different temporal scales, e.g., sub-hourly, diurnal, monthly, and annual. Understanding this variability is crucial for many wind energy modeling applications. Develop a prototype analysis tool that examines wind speed variability at diurnal and monthly temporal scales at an individual location of your choosing. Use NREL's WIND Toolkit (WTK) data set (see below) to examine hourly (60-minute interval) wind speed data for the year 2012. Use only open-source tools. Develop statistical summaries and simple visualizations that will inform the user on the wind speed variability across diurnal and monthly scales. Your solution should provide location-specific summaries but should also be developed in an extensible manner such that it could be scaled in the future to run efficiently across broad geographic extents (utilizing an HPC system or the cloud). Prepare a 15-minute section of your interview presentation on your solution and demonstrate the insight it provides. Discuss how your approach could be scaled and extended to examine temporal variability across space.

Set Up for Task

```
import numpy as np
import pandas as pd
import dateutil
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from scipy.spatial import cKDTree
import h5pyd
import h5py
```

Import Data and Subset

Data was downloaded following the instructions provided in the instruction email: https://github.com/NREL/hsds-examples/blob/master/notebooks/02_WTK_Domains_introduction.ipynb

For this to work you must first install h5pyd:

```
pip install --user h5pyd
```

Next you'll need to configure HSDS:

```
hsconfigure
```

and enter at the prompt:

```
hs_endpoint = https://developer.nrel.gov/api/hsds
```

```
hs_username = None
```

```
hs_password = None
```

```
hs_api_key = 3K3JQbjZmWctY0xmIfSYvYgtIcM3CN0cb1Y2w9bf
```

The example API key here is for demonstration and is rate-limited per IP. To get your own API key, visit []https://developer.nrel.gov/signup/

Let's import the data for the year 2012 and for the contiguous US.

```
year = '2012'
country = "conus" #
wind_file_path = "/nrel/wtk/conus/wtk_" + country + "_" + year + ".h5"
raw_h5 = h5pyd.File(wind_file_path, 'r')

list(raw_h5)
```

```
['coordinates', 'inversemoninobukhovlength_2m', 'meta', 'precipitationrate_0m', 'pressure_0m
```

The below chunk takes a look at the data dimensions. Understanding the dimensions is critical to working with .h5 files as the index numbers are what ties the separate data structures together.

```
raw_h5['coordinates']
```

```
<HDF5 dataset "coordinates": shape (2488136, 2), type "<f4">
```

```
raw_h5['meta']
```

```
<HDF5 dataset "meta": shape (2488136,), type "|V69">
```

```
raw_h5['windspeed_100m']
```

```
<HDF5 dataset "windspeed_100m": shape (8784, 2488136), type "<u2">
```

```
raw_h5['time_index']
```

```
<HDF5 dataset "time_index": shape (8784,), type "|S20">
```

So with the dimensions above we can interpret that there are almost 2.5 million coordinate points and 8,784 time points. We can then extract the wind speed at a specific time and place during the year.

Time Subset

Next we will pull out the time index data and make it more friendly for python to work with.The time data is hourly for every day of the year.

```
time_index = pd.to_datetime(raw_h5['time_index'][...].astype(str)) #save timestamp index a
time_index
```

```
DatetimeIndex(['2012-01-01 00:00:00', '2012-01-01 01:00:00',
               '2012-01-01 02:00:00', '2012-01-01 03:00:00',
               '2012-01-01 04:00:00', '2012-01-01 05:00:00',
               '2012-01-01 06:00:00', '2012-01-01 07:00:00',
               '2012-01-01 08:00:00', '2012-01-01 09:00:00',
               ...
               '2012-12-31 14:00:00', '2012-12-31 15:00:00',
               '2012-12-31 16:00:00', '2012-12-31 17:00:00',
               '2012-12-31 18:00:00', '2012-12-31 19:00:00',
               '2012-12-31 20:00:00', '2012-12-31 21:00:00',
               '2012-12-31 22:00:00', '2012-12-31 23:00:00'],
              dtype='datetime64[ns]', length=8784, freq=None)
```

```
time_df = pd.DataFrame(data = time_index, columns = ['date_time']) # save as pandas DF to

# time_Index has each hour of the day stored
```

Meta Data Subset

Next the meta data is saved. This will not be used for this model but it is helpful to understand what kind of data is embedded in the file.

```
meta = pd.DataFrame(raw_h5['meta'][...]) #save meta data as object, ellipses makes it an a
meta.head() # just used to look at the data more easily
# This has latitude and longitude columns that can be used to search for other information
```

```
    latitude   longitude  country  ... timezone elevation  offshore
0  37.603382 -127.617050  b'None'  ...       -9         0         1
1  37.620419 -127.626007  b'None'  ...       -9         0         1
2  37.637451 -127.634979  b'None'  ...       -9         0         1
3  37.654484 -127.643951  b'None'  ...       -9         0         1
4  37.671509 -127.652924  b'None'  ...       -9         0         1

[5 rows x 8 columns]
```

Coordinates Subset

```
coords = raw_h5['coordinates'][...] # save coordinates data as array, ellipses makes it an
coords_df = pd.DataFrame(data = coords)
coords_df.columns = ['latitude', 'longitude']
```

A Function to Find the Index for a Point of Interest

The function `nearest_site`, allows you to find the index numbers within a `coord_array` for a given latitude (`lat_coord`) and longitude (`lon_coord`) coordinate. The returned index numbers will allow you to find the wind speed for the point of interest in the .h5 file.

```
def nearest_site(coord_array, lat_coord, lon_coord):
    tree = cKDTree(coord_array)
    lat_lon = np.array([lat_coord, lon_coord])
    dist, pos = tree.query(lat_lon)
```

```
    return pos;
```

Testing

The below is a test for the function above for the point of interest (POI). The coordinates used in this example are for Santa Barbara, California. We are making the coords object the same as above here but it is included so that it is more obvious where things are coming from in the function call.

```
coords = raw_h5['coordinates'][...] # save coordinates data as array, ellipses makes it an
POI = (34.420830, -119.698189)

POI_indx = nearest_site(coord_array = coords,
                        lat_coord = POI[0],
                        lon_coord = POI[1])
POI_indx
```

209886

Finding the Wind Speed for `POI_indx`

For this example we will be using the data for `windspeed_100m` from the `raw_h5` extracted at the beginning of the demo. The list of available data is above under the call `list(raw_h5)`. Next the wind speed is extracted from the h5 for all time rows in the designated year at the specific location indicated by the index. It is unscaled by the `scale_factor` of 100 so that the resulting values represent m/s. Extracting this data is done by the function below `h5_extractor` that will pull the data for a desired variable from the provided coordinate point index number that is calculated above.

```
def h5_extractor(raw_h5, variable, POI_indx):
    variable_raw = raw_h5[variable]
    variable_raw = variable_raw[:, POI_indx] / variable_raw.attrs['scale_factor']
    variable = pd.DataFrame(data = variable_raw)
    return variable;
```

```
windspeed_100 = h5_extractor(raw_h5 = raw_h5,
                             variable = 'windspeed_100m',
                             POI_indx = POI_indx)
```

Make a DataFrame for Wind Speed

Next the setup work that has been done above will be made into a more useful data frame so that information can be visualized more easily. This function assumes you have completed the steps above.

```
def windspeed_point_df_maker(time_df, coords_df, windspeed, POI_indx):
    df = pd.DataFrame(dtype = float) # making a data frame with default dtype 'float'
    df['year'] = time_df['date_time'].dt.year # make a year column
    df['month'] = time_df['date_time'].dt.month # make a month column
    df['day'] = time_df['date_time'].dt.day # make a day column
    df['hour'] = time_df['date_time'].dt.hour # make an hour column
    df['latitude'] = coords_df['latitude'].iloc[POI_indx] # get all latitude coords
    df['longitude'] = coords_df['longitude'].iloc[POI_indx] # get all longitude coords
    df['windspeed'] = windspeed_100 # unscaled windspeeds
    df['avg_windspeed'] = df['windspeed'].mean() # find the average windspeed for the whol
    return df;
```

Now that we have a function to give us a more usable data frame lets make one for the wind speed at 100 meters data

```
df_100m = windspeed_point_df_maker(time_df = time_df,
                                   coords_df = coords_df,
                                   windspeed = windspeed_100,
                                   POI_indx = POI_indx)
df_100m.head()
```

	year	month	day	hour	latitude	longitude	windspeed	avg_windspeed
0	2012	1	1	0	34.414795	-119.702332	1.83	3.270203
1	2012	1	1	1	34.414795	-119.702332	0.62	3.270203
2	2012	1	1	2	34.414795	-119.702332	0.44	3.270203
3	2012	1	1	3	34.414795	-119.702332	0.29	3.270203
4	2012	1	1	4	34.414795	-119.702332	0.13	3.270203

Subset for Visualizations

The function below, `time_subset`, will allow the user to group the data into a designated subset by the hour, day, or month. Different functions are completed based on the type of plots one may want to make with the data. This function assumes that the data has been categorized as is done in the `windspeed_point_df_maker` function.

```
def time_subset(period, df):
    if period == 'hour':
        df = df.groupby(['year','month','hour','avg_windspeed'], as_index = False).agg(hou
    elif period == 'day':
        df = df.groupby(['year','month','day','avg_windspeed'], as_index = False).agg(dail
        df['date'] = pd.to_datetime(df['year'].astype(str) + '/' +df['month'].astype(str)
    elif period == 'month':
        df = df.groupby(['month', 'avg_windspeed'], as_index = False).agg(month_avg_windsp
    else:
        print("Period must be desginated as either hour, day, or month")
    return df
```

Visualizing Wind Variability

Next the data is subset to be made into visualizations. This uses the function just made `time_subset` above. First lets explore the daily summary to see average wind speed of each day.

Daily Averages

```
day_var = time_subset(period = 'day', df = df_100m)
day_var.head()
```

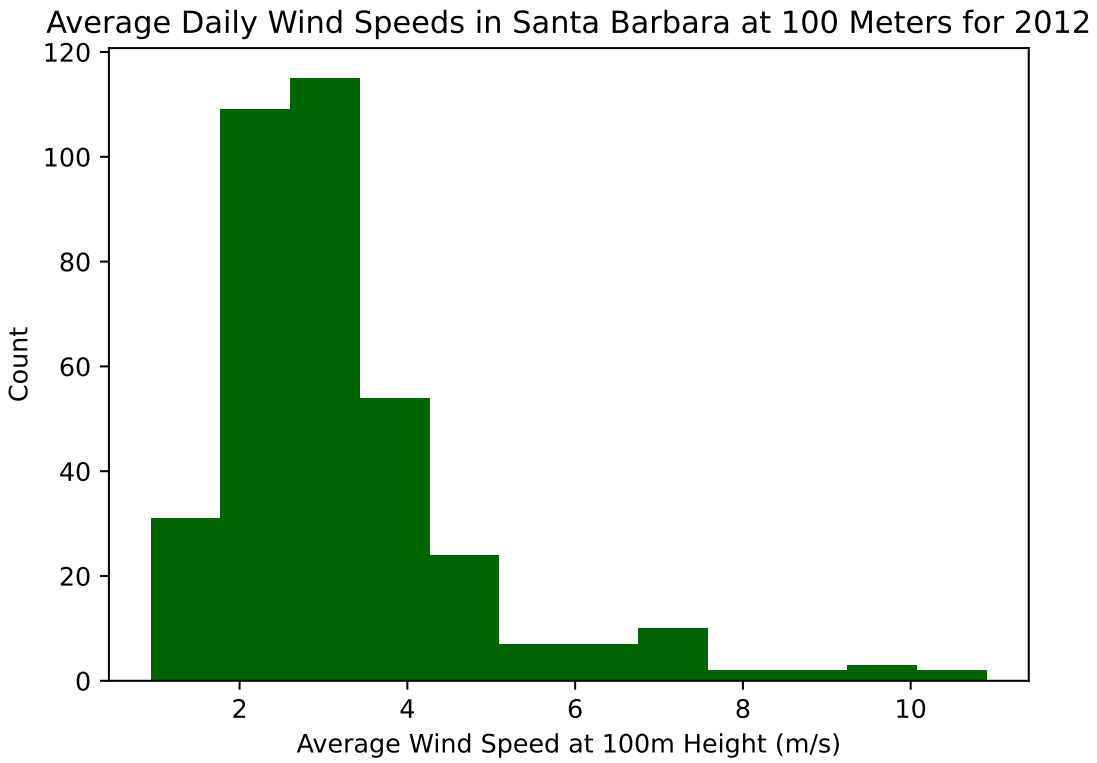
```
   year  month  day  ...  daily_avg_windspeed  standard_dev       date
0  2012      1    1  ...             0.940833      0.554593 2012-01-01
1  2012      1    2  ...             2.021667      0.985065 2012-01-02
2  2012      1    3  ...             1.599583      1.227679 2012-01-03
3  2012      1    4  ...             1.260833      0.800923 2012-01-04
4  2012      1    5  ...             1.242917      0.814827 2012-01-05

[5 rows x 7 columns]
```

First a histogram is made to understand how daily wind averages are distributed across 2012.

```
plt.clf()
plt.hist(x = day_var['daily_avg_windspeed'], color = "darkgreen", label = "Average Daily W
```

```
(array([ 31., 109., 115.,  54.,  24.,   7.,   7.,  10.,   2.,   2.,   3.,
         2.]), array([ 0.94083333,  1.77152778,  2.60222222,  3.43291667,  4.26361111,
        5.09430556,  5.925     ,  6.75569444,  7.58638889,  8.41708333,
        9.24777778, 10.07847222, 10.90916667]), <BarContainer object of 12 artists>)
```

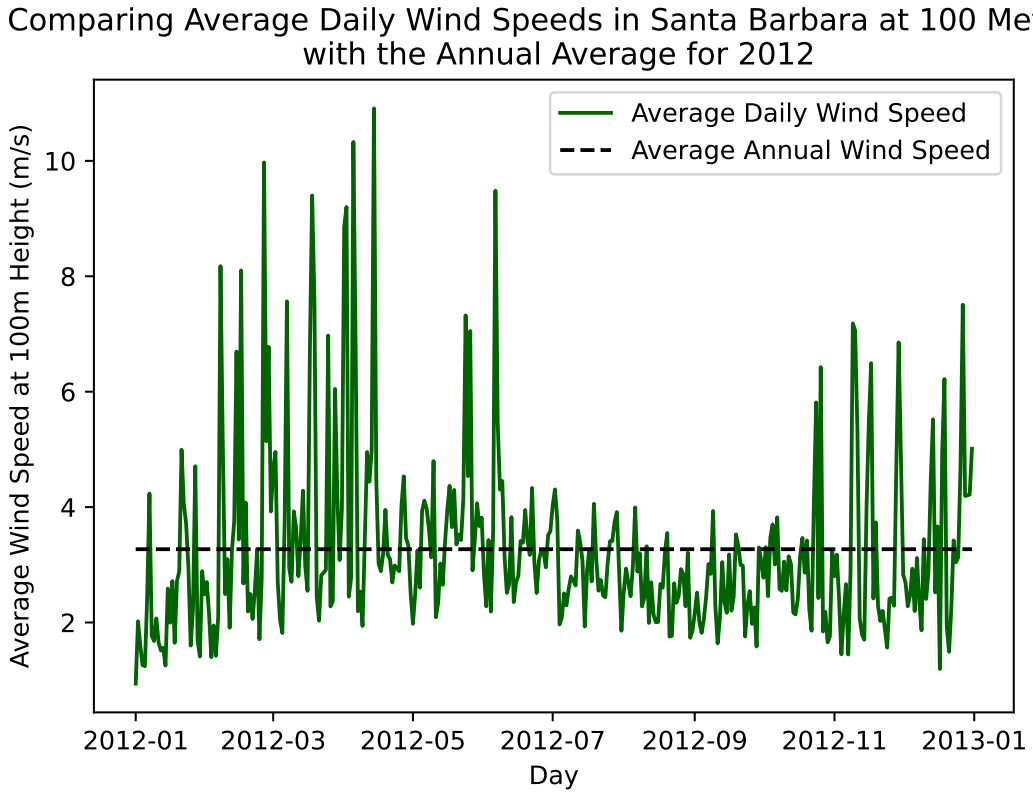
```
plt.xlabel("Average Wind Speed at 100m Height (m/s)")
plt.ylabel("Count")
plt.title("Average Daily Wind Speeds in Santa Barbara at 100 Meters for 2012")
plt.show()
```



This histogram tells us that that most days of the year have a wind speed averaging less than 5 m/s in Santa Barbara. If we look at how these variations are spread across the year are there any patterns that emerge?

```
plt.clf()
plt.plot(day_var['date'], day_var['daily_avg_windspeed'], color = "darkgreen", label = "Av
plt.plot(day_var['date'], day_var['avg_windspeed'], color = "black", label = "Average Annu
```

```
plt.xlabel("Day")
plt.ylabel("Average Wind Speed at 100m Height (m/s)")
plt.title("Comparing Average Daily Wind Speeds in Santa Barbara at 100 Meters \n with the
plt.legend()
plt.show()
```



It looks like the days with the strongest wind throughout the year are in the winter and early months however, there is a lot of variability between the high wind days and the low wind days. It appears that day to day wind variation is low from June-October. How does this look if we simplify it to month-month?

Monthly Averages

```
monthly_var = time_subset(period = 'month', df = df_100m)
monthly_var.head()
```

	month	avg_windspeed	month_avg_windspeed	standard_dev
0	1	3.270203	2.339987	1.860377
1	2	3.270203	3.702514	3.598256
2	3	3.270203	3.946653	2.837208
3	4	3.270203	4.295028	3.619121
4	5	3.270203	3.706169	2.218983

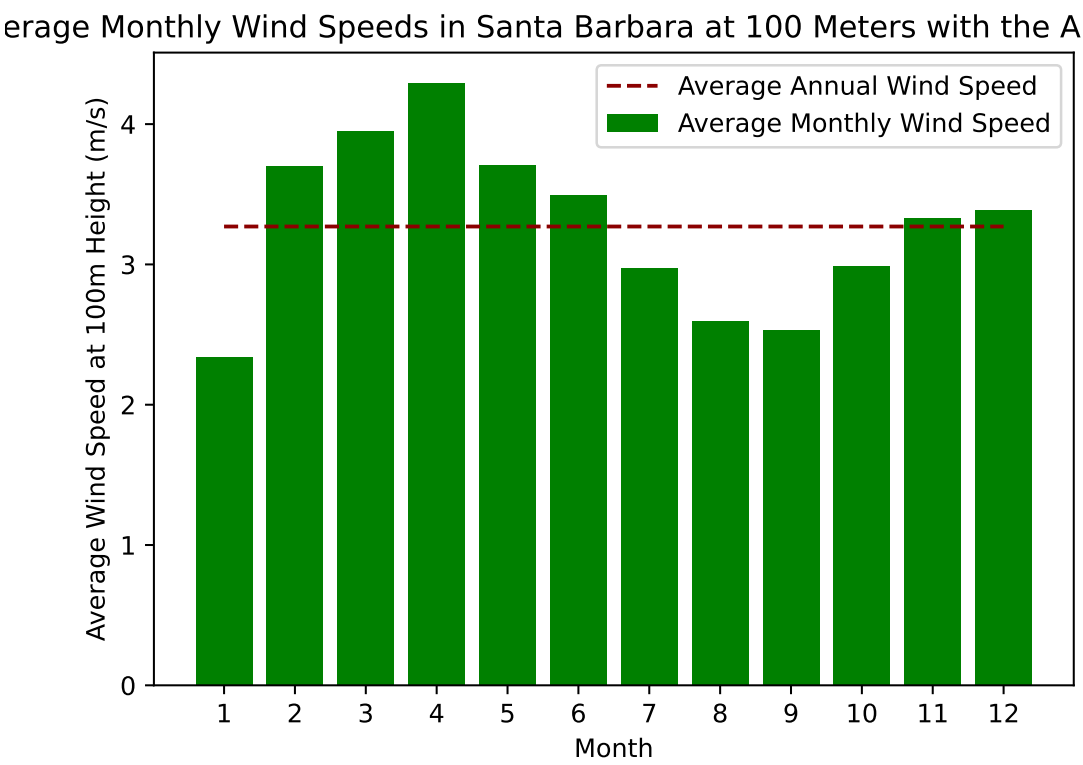
```
plt.clf()
plt.bar(monthly_var['month'], monthly_var['month_avg_windspeed'], color = "green", label =
```

```
<BarContainer object of 12 artists>
```

```
plt.xticks(monthly_var['month'])
```

```
([<matplotlib.axis.XTick object at 0x289d0a310>, <matplotlib.axis.XTick object at 0x289d0a28
```

```
plt.plot(monthly_var['month'], monthly_var['avg_windspeed'], color = "darkred", linestyle
# plt.scatter(monthly_var['month'], monthly_var['month_avg_windspeed'] + monthly_var['stan
# plt.scatter(monthly_var['month'], monthly_var['month_avg_windspeed'] - monthly_var['stan
plt.xlabel("Month")
plt.ylabel("Average Wind Speed at 100m Height (m/s)")
plt.title("Comparing Average Monthly Wind Speeds in Santa Barbara at 100 Meters with the A
plt.legend()
plt.show()
```



This confirms our observations for the day-to-day variation of wind speed for the year. The winter and early spring months are the windier months, while July-October have lower wind on average. What about the standard deviation for each month?

```
plt.clf()
plt.bar(monthly_var['month'], monthly_var['month_avg_windspeed'], color = "green", label =
```

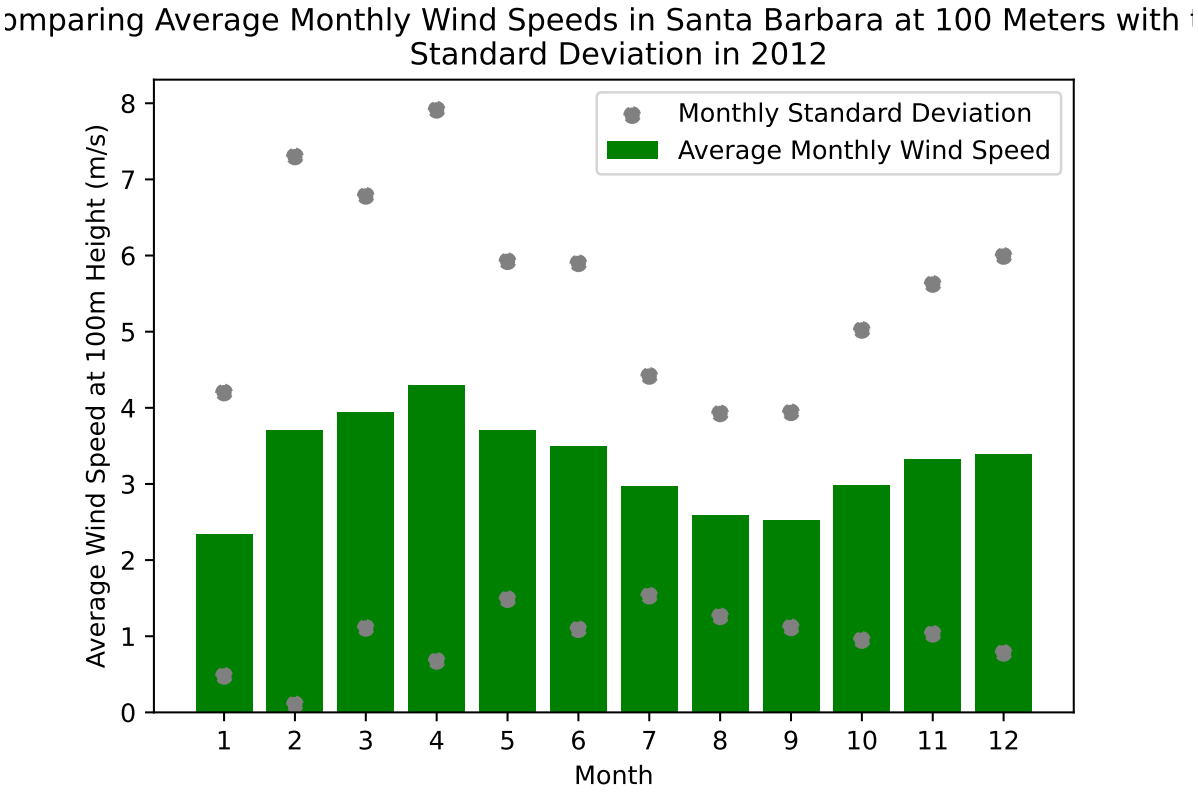
```
<BarContainer object of 12 artists>
```

```
plt.xticks(monthly_var['month'])
```

```
([<matplotlib.axis.XTick object at 0x289d676a0>, <matplotlib.axis.XTick object at 0x289d67dc
```

```
plt.scatter(monthly_var['month'], monthly_var['month_avg_windspeed'] + monthly_var['standa
plt.scatter(monthly_var['month'], monthly_var['month_avg_windspeed'] - monthly_var['standa
```

```
plt.xlabel("Month")
plt.ylabel("Average Wind Speed at 100m Height (m/s)")
plt.title("Comparing Average Monthly Wind Speeds in Santa Barbara at 100 Meters with the \
plt.legend()
plt.show()
```



Including the standard deviations in this plot show us the variability of wind in that particular month in much greater detail. So in February, you can have days with more extreme winds but you can also have days with practically no wind. Whereas in July you have a lower average wind speed, but it is less variable throughout the month. What if we wanted to look how wind speed varies for the hours in a day for a particular month?

Monthly Averages for Each Month

```
hourly_var = time_subset(period = 'hour', df = df_100m)
hourly_var.head()
```

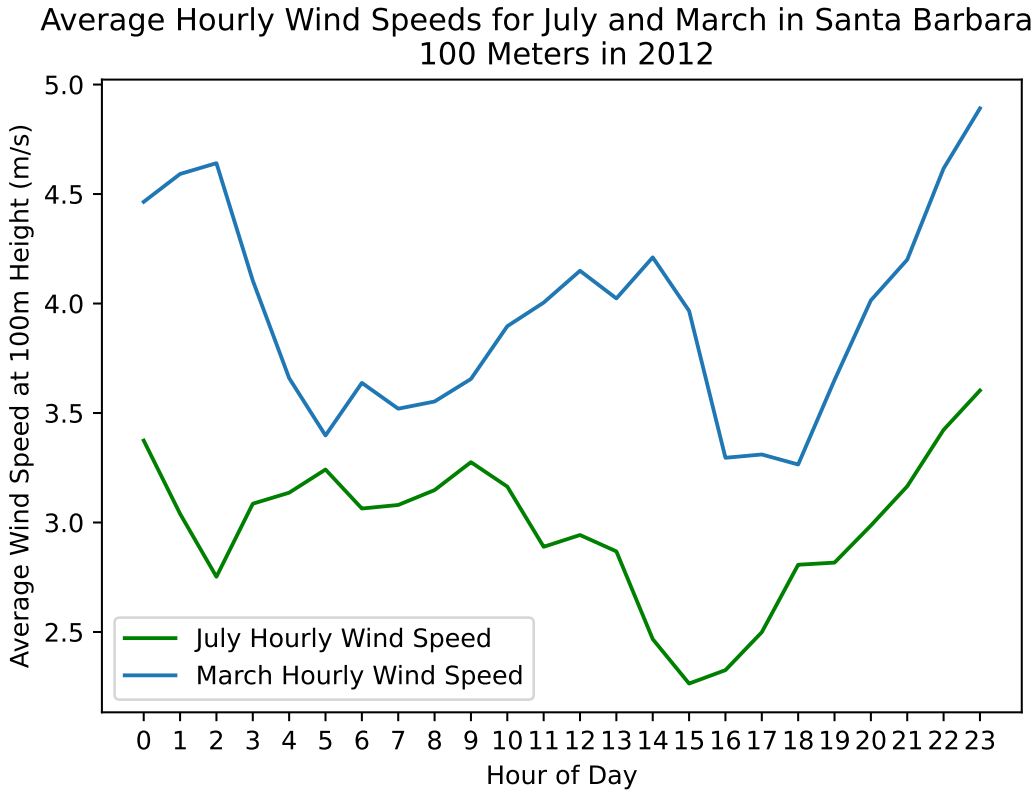
	year	month	hour	avg_windspeed	hour_avg_windspeed	standard_dev
0	2012	1	0	3.270203	2.633871	1.946564
1	2012	1	1	3.270203	2.544516	2.137604
2	2012	1	2	3.270203	2.265484	1.951186
3	2012	1	3	3.270203	2.185484	1.588781
4	2012	1	4	3.270203	1.888710	1.253342

Since we know July has less variation and March seems to have more lets compare themthan other months lets take a look there first

```
july_var = hourly_var[hourly_var['month'] == 7]
march_var = hourly_var[hourly_var['month'] == 3]

plt.clf()
plt.plot(july_var['hour'], july_var['hour_avg_windspeed'], label = "July Hourly Wind Speed
plt.xticks(july_var['hour'])
```

```
([<matplotlib.axis.XTick object at 0x289d5abb0>, <matplotlib.axis.XTick object at 0x289d5aaf
```

```
plt.plot(march_var['hour'], march_var['hour_avg_windspeed'], label = "March Hourly Wind Sp
plt.xlabel("Hour of Day")
plt.ylabel("Average Wind Speed at 100m Height (m/s)")
plt.title("Average Hourly Wind Speeds for July and March in Santa Barbara at \n 100 Meters
plt.legend()
plt.show()
```



Next Steps and Considerations

With the above code we are able to look at wind speed variation at a specific point for a given year at different heights. It is also possible to use the resulting data frames to visualize these variations. The below tries to expand the work done above to be able to compare multiple points from a data frame.

A Function to Find Indexes for Multiple Points of Interest Indices

The function below, `multiple_site_index`, works with the `nearest_site` function above to work with multiple coordinate points rather than one location. The function assumes that the data frame's first column is "lat" and the second column is "lon".

```
def multiple_site_index(POI_df, coord_array):
    POI_df.columns = ['lat', 'lon']
    results = pd.DataFrame(dtype = float)
    for ind in POI_df.index:
```

```
        lat = POI_df['lat'][ind]
        lon = POI_df['lon'][ind]
        POI_indx = nearest_site(coord_array, lat, lon)
        results[ind](POI_indx)
    return results
```

```
coords = raw_h5['coordinates'][...] # save coordinates data as array, ellipses makes it an

POI_df_test = coords_df.head()

# POI_multi = multiple_site_index(POI_df = POI_df_test, coord_array = coords)
# POI_multi
```

```
coords_df = pd.DataFrame(data = coords)
coords_df.columns = ['latitude', 'longitude']
# coords_df = coords_df.astype(float)


# find the max and min of the dataset to create bounding limits
lat_max = coords_df['latitude'].max()
lat_min = coords_df['latitude'].min()
long_max = coords_df['longitude'].max()
long_min = coords_df['longitude'].min()
```

```
renv::snapshot()
```

```
* Lockfile written to '~/Documents/Jobs/NREL/NREL_task_2022/renv.lock'.
* Python requirements are already up to date.
```

```
# renv::restore()
```