Notebook

April 29, 2025

USER GUIDE (DOCUMENTATION)

EXPLORING HURRICANE FRANCINE

GRP 8781 [Team member 1: ***Halkano Molu Guracha***, Team member 2: ***Biswajit Palit***, Team member 3: ***Joel Muchemeranwa***]

PROBLEM SET PART 2, QUESTION 5 & 6: SAMPLE CODE ON HOW TO USE:

a) Python code to import and structure into useful data structures

b) Exploratory data analysis of sample data

This user guide explores Hurricane Francine's path and potential impact using geospatial data analysis within a financial engineering context. In further exploration, the user can leverage Python libraries like geopandas and folium to visualize the hurricane's trajectory and overlay it with relevant financial infrastructure or assets. The insights gained from this analysis can inform risk assessment models, insurance pricing, and investment strategies related to weather-sensitive financial instruments or infrastructure. This approach is particularly valuable in understanding how extreme weather events like Hurricane Francine can impact the financial landscape and to what extent.

0.1 7.1 Load Python Packages

There are a few packages we will need for this demonstration.

[1]:
```
!pip install fiona
```

```
Collecting fiona
  Downloading
fiona-1.10.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(56 kB)
                            56.6/56.6 kB
1.8 MB/s eta 0:00:00
Requirement already satisfied: attrs>=19.2.0 in
/usr/local/lib/python3.11/dist-packages (from fiona) (25.3.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-
packages (from fiona) (2025.1.31)
Requirement already satisfied: click~=8.0 in /usr/local/lib/python3.11/dist-
packages (from fiona) (8.1.8)
Collecting click-plugins>=1.0 (from fiona)
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl.metadata (6.4 kB)
```

```
Collecting cligj>=0.5 (from fiona)
  Downloading cligj-0.7.2-py3-none-any.whl.metadata (5.0 kB)
Downloading
fiona-1.10.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3
MB)
                         17.3/17.3 MB
33.4 MB/s eta 0:00:00
Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Installing collected packages: cligj, click-plugins, fiona
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.10.1
```

[2]:

```
!pip install geopandas
```

```
Requirement already satisfied: geopandas in /usr/local/lib/python3.11/dist-
packages (1.0.1)
Requirement already satisfied: numpy>=1.22 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (2.0.2)
Requirement already satisfied: pyogrio>=0.7.2 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (0.10.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from geopandas) (24.2)
Requirement already satisfied: pandas>=1.4.0 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (2.2.2)
Requirement already satisfied: pyproj>=3.3.0 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (3.7.1)
Requirement already satisfied: shapely>=2.0.0 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (2.1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas>=1.4.0->geopandas)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas>=1.4.0->geopandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas>=1.4.0->geopandas) (2025.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-
packages (from pyogrio>=0.7.2->geopandas) (2025.1.31)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas>=1.4.0->geopandas) (1.17.0)
```

[47]:

```
!pip install install-jdk
```

```
Requirement already satisfied: install-jdk in /usr/local/lib/python3.11/dist-
packages (1.1.0)
```

[5]:

```
import jdk
from jdk.enums import OperatingSystem, Architecture
```

```
jdk.install('11', operating_system=OperatingSystem.LINUX)
```

[5]:
```
'/root/.jdk/jdk-11.0.27+6'
```

[6]:
```
import os
jdk_version = 'jdk-11.0.25+9' #change with your version
os.environ['JAVA_HOME'] = '/root/.jdk/jdk-11.0.25+9'
os.environ['PATH'] = f"{os.environ.get('PATH')}:{os.environ.get('JAVA_HOME')}/
 ↪bin"
```

[7]:
```
# Import packages for this application
# pandas is used to process dataframe and geopandas is used to process␣
 ↪geodataframe
# fiona is used to read or write various formats of geospatial data
# urllib is used to pull data on the internet using url
#zipfile is used to unzip a zipped file
import os
import pandas as pd
import geopandas as gpd
import fiona
import urllib.request
import zipfile
```

0.2 7.2 Data for Hurricane Francine

The hurricane path data is in a PDF file. As such, we need to download the tabula package to handle PDF files.

[46]:
```
!pip install tabula-py
```

```
Requirement already satisfied: tabula-py in /usr/local/lib/python3.11/dist-
packages (2.10.0)
Requirement already satisfied: pandas>=0.25.3 in /usr/local/lib/python3.11/dist-
packages (from tabula-py) (2.2.2)
Requirement already satisfied: numpy>1.24.4 in /usr/local/lib/python3.11/dist-
packages (from tabula-py) (2.0.2)
Requirement already satisfied: distro in /usr/local/lib/python3.11/dist-packages
(from tabula-py) (1.9.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.25.3->tabula-py)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas>=0.25.3->tabula-py) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas>=0.25.3->tabula-py) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas>=0.25.3->tabula-py) (1.17.0)
```

[9]:

```
# Import tabula package for PDF handling
import tabula
```

Next, let's retrieve the PDF file from the National Hurricane Center website.

[12]:

```
# Retrieve PDF file from NHC website
#url = "https://www.nhc.noaa.gov/data/tcr/AL092011_Francine.pdf"
url = "https://www.nhc.noaa.gov/data/tcr/AL062024_Francine.pdf"

francine_pdf, _ = urllib.request.urlretrieve(url)
```

Then, we will use a method from the tabula package to convert a PDF file to a csv file.

[14]:

```
# Convert PDF file to csv file, and then a pandas dataframe
tabula.convert_into(francine_pdf, "francine.csv", output_format="csv",␣
 ↪stream=True, pages = 12)
francine_1 = pd.read_csv("francine.csv")
francine_1
```

[14]:

```
    Date/Time Latitude Longitude Pressure        Wind          Unnamed: 5
0       (UTC)     (°N)      (°W)     (mb)  Speed (kt)               Stage
1   08 / 1800     21.4      94.5     1003          45                 low
2   09 / 0000     22.0      94.8     1003          45                   "
3   09 / 0600     22.7      95.1     1003          45                   "
4   09 / 1200     23.2      95.5     1002          45      tropical storm
5   09 / 1800     23.7      95.9      996          50                   "
6   10 / 0000     23.9      96.1      992          55                   "
7   10 / 0600     24.1      96.2      992          55                   "
8   10 / 1200     24.5      95.8      990          55                   "
9   10 / 1800     25.3      95.3      987          55                   "
10  11 / 0000     26.0      94.7      982          65           hurricane
11  11 / 0600     26.6      94.1      977          80                   "
12  11 / 1200     27.5      93.2      976          80                   "
13  11 / 1800     28.6      92.1      974          85                   "
14  11 / 2200     29.3      91.3      972          90                   "
15  12 / 0000     29.6      90.9      980          70                   "
16  12 / 0600     30.5      90.3      988          45      tropical storm
17  12 / 1200     31.8      90.1      995          30 tropical depression
18  12 / 1800     33.4      89.7      996          25       extratropical
19  13 / 0000     34.7      90.5      998          25                   "
20  13 / 0600     35.4      90.9     1001          20                   "
21  13 / 1200     35.9      91.5     1005          15                   "
22  13 / 1800     35.7      92.0     1009          15                   "
```

From the last code output, we can see that the second row does not contain data values. They are measurement units/information for each variable. For example, the measurement unit for wind speed is knots (kt). Also, the last row does not have any data values. We are going to drop these two rows.

[16]:

```
# Drop rows with measurement units or no data values
francine_1.drop([0,22], inplace = True)
francine_1
```

[16]:

```
    Date/Time Latitude Longitude Pressure Wind           Unnamed: 5
1   08 / 1800     21.4      94.5     1003   45                  low
2   09 / 0000     22.0      94.8     1003   45                    "
3   09 / 0600     22.7      95.1     1003   45                    "
4   09 / 1200     23.2      95.5     1002   45       tropical storm
5   09 / 1800     23.7      95.9      996   50                    "
6   10 / 0000     23.9      96.1      992   55                    "
7   10 / 0600     24.1      96.2      992   55                    "
8   10 / 1200     24.5      95.8      990   55                    "
9   10 / 1800     25.3      95.3      987   55                    "
10  11 / 0000     26.0      94.7      982   65            hurricane
11  11 / 0600     26.6      94.1      977   80                    "
12  11 / 1200     27.5      93.2      976   80                    "
13  11 / 1800     28.6      92.1      974   85                    "
14  11 / 2200     29.3      91.3      972   90                    "
15  12 / 0000     29.6      90.9      980   70                    "
16  12 / 0600     30.5      90.3      988   45       tropical storm
17  12 / 1200     31.8      90.1      995   30  tropical depression
18  12 / 1800     33.4      89.7      996   25        extratropical
19  13 / 0000     34.7      90.5      998   25                    "
20  13 / 0600     35.4      90.9     1001   20                    "
21  13 / 1200     35.9      91.5     1005   15                    "
```

Next, let's convert numeric variables to float types.

[19]:

```
# Correct data types for numeric variables
convert_dict = {'Date/Time': str,
                'Latitude': float,
                'Longitude': float,
                'Pressure': float,
                'Wind': float
                }

francine_1 = francine_1.astype(convert_dict)
print(francine_1.dtypes)
```

```
Date/Time      object
Latitude      float64
Longitude     float64
Pressure      float64
Wind          float64
Unnamed: 5     object
dtype: object
```

0.3 7.3 Creating the Date-Time Variable

Next, we can see that the date/time variable does not contain month and year information. Therefore, we're going to create a new date/time variable to provide complete date/time information.

```
[21]: # Create a new Date Time variable to contain month and year information
      francine_1[["Date","Time"]] = francine_1["Date/Time"].str.split(" / ", expand =␣
       ↪True)
      francine_1["Date_Time"] = "08/" + francine_1["Date"] + "/2024/" +␣
       ↪francine_1["Time"]
      francine_1.set_index("Date_Time")
      francine_1
```

```
[21]:     Date/Time  Latitude  Longitude  Pressure  Wind          Unnamed: 5 Date  \
      1   08 / 1800      21.4       94.5    1003.0  45.0                 low   08
      2   09 / 0000      22.0       94.8    1003.0  45.0                   "   09
      3   09 / 0600      22.7       95.1    1003.0  45.0                   "   09
      4   09 / 1200      23.2       95.5    1002.0  45.0      tropical storm   09
      5   09 / 1800      23.7       95.9     996.0  50.0                   "   09
      6   10 / 0000      23.9       96.1     992.0  55.0                   "   10
      7   10 / 0600      24.1       96.2     992.0  55.0                   "   10
      8   10 / 1200      24.5       95.8     990.0  55.0                   "   10
      9   10 / 1800      25.3       95.3     987.0  55.0                   "   10
      10  11 / 0000      26.0       94.7     982.0  65.0           hurricane   11
      11  11 / 0600      26.6       94.1     977.0  80.0                   "   11
      12  11 / 1200      27.5       93.2     976.0  80.0                   "   11
      13  11 / 1800      28.6       92.1     974.0  85.0                   "   11
      14  11 / 2200      29.3       91.3     972.0  90.0                   "   11
      15  12 / 0000      29.6       90.9     980.0  70.0                   "   12
      16  12 / 0600      30.5       90.3     988.0  45.0      tropical storm   12
      17  12 / 1200      31.8       90.1     995.0  30.0  tropical depression   12
      18  12 / 1800      33.4       89.7     996.0  25.0        extratropical   12
      19  13 / 0000      34.7       90.5     998.0  25.0                   "   13
      20  13 / 0600      35.4       90.9    1001.0  20.0                   "   13
      21  13 / 1200      35.9       91.5    1005.0  15.0                   "   13

          Time        Date_Time
      1   1800  08/08/2024/1800
      2   0000  08/09/2024/0000
      3   0600  08/09/2024/0600
      4   1200  08/09/2024/1200
      5   1800  08/09/2024/1800
      6   0000  08/10/2024/0000
      7   0600  08/10/2024/0600
      8   1200  08/10/2024/1200
      9   1800  08/10/2024/1800
      10  0000  08/11/2024/0000
      11  0600  08/11/2024/0600
```

```
12  1200  08/11/2024/1200
13  1800  08/11/2024/1800
14  2200  08/11/2024/2200
15  0000  08/12/2024/0000
16  0600  08/12/2024/0600
17  1200  08/12/2024/1200
18  1800  08/12/2024/1800
19  0000  08/13/2024/0000
20  0600  08/13/2024/0600
21  1200  08/13/2024/1200
```

0.4 7.4 Ensuring Spatial Data is Correct

We also notice that the data values for longitude in the dataset are all positive. However, the direction for longitude is "W." The numbers for longitude should all be negative based on Figure 1 in section 3. Hence, we need to add a negative sign in front of all numbers for the longitude variable to correctly reflect the location.

[22]:

```
# Adjust Longitude value to correctly reflect the geolocation
francine_1['Longitude'] = 0 - francine_1['Longitude']
```

[26]:

```
# Select the variables we are interest for next steps
francine_2 = francine_1[["Date_Time","Longitude","Latitude","Wind"]]
```

0.5 7.5 Converting to a Geodataframe

Now we have a good dataframe for Hurricane Francine's path. To convert this dataframe to a geospatial dataframe, we need to create a geometry variable, as explained in the previous section. We will use a method from geopandas to create this variable. One of the parameters in the following code is "EPSG:4326", which is the code name for the latitude-longitude system we are familiar with. Remember there are several CRS systems available, but we'll proceed with this commonly used one.

[27]:

```
# Create a geolocation variable
geometry = gpd.points_from_xy(francine_2.Longitude, francine_2.Latitude,␣
 ↪crs="EPSG:4326")
```

Now let's convert the current pandas dataframe to a geodataframe.

[28]:

```
# Convert the current dataframe to a geodataframe with geometry variable
francine_3 = gpd.GeoDataFrame(
    francine_2, geometry=geometry, crs="EPSG:4326"
)
```

Great! Now we have a geodataframe. Let's check out the first five entries in this dataframe.

[29]:

```
francine_3.head()
```

[29]:

```
         Date_Time  Longitude  Latitude  Wind             geometry
1  08/08/2024/1800      -94.5      21.4  45.0  POINT (-94.5 21.4)
```

```
2  08/09/2024/0000      -94.8      22.0  45.0    POINT (-94.8 22)
3  08/09/2024/0600      -95.1      22.7  45.0  POINT (-95.1 22.7)
4  08/09/2024/1200      -95.5      23.2  45.0  POINT (-95.5 23.2)
5  08/09/2024/1800      -95.9      23.7  50.0  POINT (-95.9 23.7)
```

Under the geometry variable in the above geodataframe, we have point shapes and their coordinate pairs on the map. Let's confirm the type of our new dataframe.

[30]:
```
type(francine_3)
```

[30]:
```
geopandas.geodataframe.GeoDataFrame
```

And let's check our geometry variable.

[31]:
```
type(francine_3['geometry'])
```

[31]:
```
geopandas.geoseries.GeoSeries
```

The geodataframe basically behaves like the pandas dataframe. Therefore, we can apply the same methods and analysis from pandas to geopandas. Here is one example.

[33]:
```
print("Mean wind speed of Hurricane Francine is {} knots and it can go up to {}␣
 ↪knots maximum".format(round(francine_2['Wind'].mean(),4),
                                                                             ␣
 ↪         francine_2['Wind'].max())+".")
```

```
Mean wind speed of Hurricane Francine is 51.4286 knots and it can go up to 90.0
knots maximum.
```

0.5.1 7.6 Data for U.S. State Borders

Before we draw Hurricane Francine's path on a map, we would like to add a layer with U.S. state borders to the map. With the state border visualization along with Hurricane Francine's path, we can learn which states were affected by this hurricane. We will pull the state border file from the United States Census Bureau's website. This is a zipped shapefile. We first need to import a package to unzip the file.

[34]:
```
# Import a file unzip package
from zipfile import ZipFile
```

[36]:
```
# Retrieve US State Shapfile from United States Census Bureau
#us_state_url = "        http://www2.census.gov/geo/tiger/TIGER2012/STATE/
 ↪tl_2012_us_state.zip"
us_state_url = "https://www.nhc.noaa.gov/gis/best_track/al092024_best_track.zip"

us_state_shape_zip, _ = urllib.request.urlretrieve(us_state_url)
```

[37]:
```
# Unzip the zipped shapefile and assign it a new file name "us_state_shape"
with ZipFile(us_state_shape_zip, 'r') as zObject:
    zObject.extractall("us_state_shape")
```

Once we unzip the file, we will use the read_file method from geopandas to read this file as a geodataframe for further data processing.

[38]:
```
# Read in our shapefile to a geodataframe
us_state_shape_g = gpd.read_file("us_state_shape")
```

```
/usr/local/lib/python3.11/dist-packages/pyogrio/geopandas.py:265: UserWarning:
More than one layer found in 'us_state_shape': 'AL092024_lin' (default),
'AL092024_radii', 'AL092024_windswath', 'AL092024_pts'. Specify layer parameter
to avoid this warning.
  result = read_func(
```

[39]:
```
# Check the metadata of the new geodataframe
us_state_shape_g.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   STORMNUM   10 non-null     float64
 1   STORMTYPE  10 non-null     object
 2   SS         10 non-null     float64
 3   geometry   10 non-null     geometry
dtypes: float64(2), geometry(1), object(1)
memory usage: 452.0+ bytes
```

[40]:
```
# Check a few entries of the new geodataframe
us_state_shape_g.head()
```

[40]:
```
   STORMNUM STORMTYPE   SS                                          geometry
0       9.0        DB  0.0  LINESTRING (-81.7 17.2, -81.9 17.8, -82.2 18.2…
1       9.0        TS  0.0  LINESTRING (-83.7 19.2, -84.6 19.4, -85.2 19.7…
2       9.0        HU  1.0  LINESTRING (-86.2 21.1, -86.5 22, -86.7 22.8, …
3       9.0        HU  2.0                LINESTRING (-85.8 24.7, -85 26.6)
4       9.0        HU  3.0                LINESTRING (-85 26.6, -84.3 28.7)
```

0.5.2 7.7 Geospatial Data Visualization

Now we have a file for Hurricane Francine's path and a file for U.S. state borders. They are also both in geodataframe forms. We can put all the information in one map for visualization. We will use the folium package to draw the map and add the U.S. state borders and the hurricane's path to the map.

[45]:
```
!pip install folium
```

```
Requirement already satisfied: folium in /usr/local/lib/python3.11/dist-packages
(0.19.5)
```

```
Requirement already satisfied: branca>=0.6.0 in /usr/local/lib/python3.11/dist-
packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.11/dist-
packages (from folium) (3.1.6)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from folium) (2.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in /usr/local/lib/python3.11/dist-
packages (from folium) (2025.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2>=2.9->folium) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->folium) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->folium) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->folium) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->folium) (2025.1.31)
```

[42]:

```python
# Import a mapping library
import folium
```

Great! Now it's time to put the information on a map.

[44]:

```python
# Draw Hurricane Francine's path and other infomation to a map

# First, create a basemap
map = folium.Map(location=[30,-102], zoom_start=4, control_scale=True)

# Then add the first layer of US state borders to the map
folium.GeoJson(us_state_shape_g).add_to(map)

# Then add the hurricane travel path to the map. We use a red dot to represent␣
 ↪the hurricane's location at a specific date/time. Then we add an information␣
 ↪box and a popup box. If you hoover your mouse cursor to the red dot, the map␣
 ↪will show you date/time linked to the location and the wind speed.
folium.GeoJson(francine_3,
               marker=folium.Circle(radius=2000, fill_color="red",␣
 ↪fill_opacity=0.4, color="red", weight=5),
              tooltip=folium.GeoJsonTooltip(fields=["Date_Time","Wind"]),
              popup=folium.GeoJsonPopup(fields=["Date_Time","Wind"]),).
 ↪add_to(map)

map
```

[44]:

```
<folium.folium.Map at 0x795a197bdd50>
```

Voila! We just created a map overlayed with U.S. state borders and Hurricane Francine's path. In the upper left corner of the map, there is an icon you can use to zoom in and out. We see U.S. state borders in solid blue lines on the map. Hurricane Francine's path is represented by a series of red dots on the map. When you hover your cursor over one of the red dots, an information box will show up, providing date/time and wind speed information. With this map, we can see Hurricane Francine path through different states. Now, you can easily find out the velocity at different points in time and specific locations?.

0.6 8. Conclusion

In this User Guide, we demonstrated a simple geospatial data application using Python. Through this application, we learned how to process different types of geospatial data. We also learned how to overlay processed data onto a map for data visualization.

0.6.1 References

1. Awati, Rahul. "Latitude and Longitude." Informa TechTarget, August 2022. https://www.techtarget.com/whatis/definition/latitude-and-longitude.

This notebook was converted with convert.ploomber.io