Lab 02 - Sifting Weather Data

Prelab

Complete Example 3.3 (p. 89) from the textbook. Coding this example will help you with the lab assignment. You can find the relevant .xlsx-file posted along with these lab instructions on D2L. After completing Example 3.3, save your script in a Jupyter notebook. Your work completing Example 3.3 does not need to be submitted for credit.

Lab

Import the weather data from the from the `weather\_data.xls` file into a Panda's dataframe called `wdf`. This time, the dataframe will contain a couple values equal to `-99999`. Transpose the data (as done in Example 3.3) so the data for each month is in columns. Save the transposed dataframe to the varaiable `df`. Complete these steps using the follwoing commands.

```
import numpy as np
import pandas as pd
wdf = pd.read_excel('weather_data.xlsx', header=None)
df = wdf.transpose()
```

Search the Pandas pacakge documentation help section to see how to use the `pd.read_excel()` function words and why we need the keyword argument `header=None` in this example.

Note: the .xlsx-file must be in your current file path, and quotes need to surround the file name.

```
In [1]: import numpy as np
        import pandas as pd
        wdf = pd.read_excel('weather_data.xlsx', header=None)
        df = wdf.transpose()
```

Afterwards, running `print(df.shape)` will indicate the dataframe `df` has dimensions of 31 rows x 12 columns. Also check the value in row 0, column 3 in the dataframe is `260` by running the `df.head()` method.

```
In [2]: print(df.shape)
        df.head()
```

```
(31, 12)
```

```
Out[2]:     0    1   2    3   4   5   6   7   8    9   10  11
        0    0   61   2  260  47   0   0   0   0    0    1   0
        1    0  103   0    1   0   0   0  45   0    0  163   0
        2  272    0  17    0   0  30   0   0   0    0    5   0
        3    0    2  27    0   0  42   0   0   0   14    0   0
        4    0    0   0    0   5   0   0   0   0  156    0   0
```

Use the following line to change all of the values in the ndataframe `wd` which are equal to `-99999` and set them to `NaN` (not a number).

```
df[df==-99999]=np.nan
```

Use the `df.tail()` method to view the last few rows in the dataframe. We see row 30, column 5 has the value '`NaN` (not a number).

```
In [13]: df[df==-99999]=np.nan
         df.tail()
```

```
Out[13]:      0     1   2     3   4     5   6   7      8   9    10  11
         26   0   0.0  78  35.0   2  35.0   0   0    6.0   0   2.0   0
         27   0  62.0   0  13.0   0  14.0   5   0  138.0   0   0.0   0
         28   0   NaN   5  86.0   0  14.0   0   0   58.0   0   0.0   0
         29  33   NaN   8   0.0   0   8.0   0   0   10.0   0   0.0   0
         30  33   NaN   0   NaN   0   NaN   0   0    NaN   1   NaN   0
```

By completing Example 3.3, you know the data in the excel file represents rainfall in *hundredths of an inch*, but for brevity's sake, this lab will treat the data as if it were rainfall in *inches*. There is no need to change or scale the data before further processing.

Define a variable called `month` and set it equal to `1` for now. This will represent the month for which we will compute statistics. Later, you will change `month` to other values. `0` means January, `1` means February, and so on.

Use the value in `month` to extract that month's data from your `wd` dataframe. Store these values in a new list called `m`. Because January has 31 days, you only see values in **m** that are positive or zero (no values of -99999). Since February has 28 days, you will see positive numbers, zeros and `NaN` present in the list.

For example, according to our data, the rainfall for the first 3 days in January are 0 inches, 0 inches, and 272 inches. The average rain fall per day in the first 3 days of January is 90.667. The average rain fall for the days of January 2nd and 3rd is 136.

Compute and store the following statistics/properties of the data in the dataframe `df1` in variables named as follows:

- `m_avg`: average inches fallen in month `m`
- `m_max`: maximum inches fallen in month `m`
- `d_max`: day(s) of month `m` in which `m_max` inches fell
- `nd_zero`: number of day(s) of month `m` in which zero inches fell
- `d_zero`: day(s) of month `m` in which zero inches fell
- `d_nonzero`: day(s) of month `m` in which at least some rain fell
- `nd_nonzero`: number of day(s) in month `m` in which at least some rain fell

- `m_avg_nonzero`: average inches of only day(s) in which some rain fell

```
In [49]: m = 1


         m_avg = np.mean(df[m])
         print(m_avg)

         m_max = np.max(df[m])
```

```
    print(m_max)

    d_max = df.index[df[m] == m_max]
    print(d_max[0])

    nd_zero = sum(df[m]==m_max)
    print(nd_zero)

    nd_nonzero = sum(df[m]>0)
    print(d_nonzero)

    df_list = list(df[m].values)
    pos_rain_lst = [x for x in df_list if x>0 ]
    np.mean(pos_rain_lst)
```

```
18.392857142857142
135.0
17
1
14
```

```
Out[49]: 36.785714285714285
```

Test all of your values for the month of January manually to make sure they are correct.

Run your script for the February data (`m=1`) and check the results against manual calculations for that month.

You may wish to test your script against other month's data. When you are convinced your results and variable names are correct, Restart the notebook Kernal and clear all output. Run each cell one last time and ensure the script is error free.

For your own records, save your lab2.ipnb file and your `weather_data.xlsx` file to the same directory. You do not need to submit your `weather\_data.xlsx` file because your instructor will have their own copy to test your script.

Deliverable

One .ipynb-file with output all cells run. The .ipynb-file should run without errors. Uploaded the .ipynb file to D2L by the end of the lab period.

By D. Kruger, adapted by P. Kazarinoff, Portland Community College, 2018