8-1-aggregating-pandas-dataframes

March 28, 2024

1 Hands-on Activity 8.1: Aggregating Data with Pandas

1.0.1 8.1.1 Intended Learning Outcomes

After this activity, the student should be able to:

- Demonstrate querying and merging of dataframes
- Perform advanced calculations on dataframes
- Aggregate dataframes with pandas and numpy
- Work with time series data

1.0.2 8.1.2 Resources

- Computing Environment using Python 3.x
- Attached Datasets (under Instructional Materials)

1.0.3 8.1.3 Procedures

The procedures can be found in the canvas module. Check the following under topics:

- 8.1 Weather Data Collection
- 8.2 Querying and Merging
- 8.3 Dataframe Operations
- 8.4 Aggregations
- 8.5 Time Series

1.0.4 8.1.4 Data Analysis

Based on the results of the data, to summarize each procedure: using query was about filtering specific values from the data set. An aggregate function groups the values into a single summary value that perform basic data analysis tool. Using window rolling, it take a window of n where it will only perform operations on that window n. While pivot_table manipulates the position of the values. Lastly the merge function where it will literally merge the two data frame.

1.0.5 8.1.5 Supplementary Activity

Using the CSV files provided and what we have learned so far in this module complete the following exercises:

[ ]:
```
import pandas as pd
import numpy as np
```

```python
file_path = '/content/earthquakes.csv'
ffile_path = '/content/faang.csv'

erthdf = pd.read_csv(file_path)
fngdf = pd.read_csv(ffile_path)

erthdf.head()
```

[ ]:

```
    mag magType           time                place  tsunami parsed_place
0  1.35      ml  1539475168010   9km NE of Aguanga, CA        0   California
1  1.29      ml  1539475129610   9km NE of Aguanga, CA        0   California
2  3.42      ml  1539475062610   8km NE of Aguanga, CA        0   California
3  0.44      ml  1539474978070   9km NE of Aguanga, CA        0   California
4  2.16      md  1539474716050  10km NW of Avenal, CA        0   California
```

[ ]:

```python
fngdf.head()
```

[ ]:

```
  ticker        date    open    high       low   close    volume
0     FB  2018-01-02  177.68  181.58  177.5500  181.42  18151903
1     FB  2018-01-03  181.88  184.78  181.3300  184.67  16886563
2     FB  2018-01-04  184.90  186.21  184.0996  184.33  13880896
3     FB  2018-01-05  185.59  186.90  184.9300  186.85  13574535
4     FB  2018-01-08  187.20  188.90  186.3300  188.28  17994726
```

1. With the earthquakes.csv file, select all the earthquakes in Japan with a magType of mb and a magnitude of 4.9 or greater.

[ ]:

```python
erthdf_mag = erthdf.query('magType == "mb" and mag > 4.9')
erthdf_mag
```

[ ]:

```
      mag magType           time                                 place  \
227   5.2      mb  1539389603790                  15km WSW of Pisco, Peru
258   5.1      mb  1539380306940            236km NNW of Kuril'sk, Russia
391   5.1      mb  1539337221080                  Pacific-Antarctic Ridge
623   5.1      mb  1539270492120       162km S of Severo-Kuril'sk, Russia
719   5.2      mb  1539236855840       163km S of Severo-Kuril'sk, Russia
…     …       …              …                                       …
9017  5.0      mb  1537304303130     181km NE of Raoul Island, New Zealand
9175  5.2      mb  1537262729590             126km N of Dili, East Timor
9176  5.2      mb  1537262656830       90km S of Raoul Island, New Zealand
9213  5.1      mb  1537255481060                          South of Tonga
9304  5.1      mb  1537236235470  34km NW of Finschhafen, Papua New Guinea

      tsunami          parsed_place
227         0                  Peru
258         0                Russia
```

```
391          0  Pacific-Antarctic Ridge
623          0                   Russia
719          0                   Russia
…        …                        …
9017         0              New Zealand
9175         1              East Timor
9176         0              New Zealand
9213         0                    Tonga
9304         1         Papua New Guinea

[80 rows x 6 columns]
```

2. Create bins for each full number of magnitude (for example, the first bin is 0-1, the second is 1-2, and so on) with a magType of ml and count how many are in each bin.

```
[ ]: # Get the magType ml only
erth_ml = erthdf[erthdf['magType'] == 'ml']

# Create bins
bins = [i for i in range(int(erth_ml['mag'].min()), int(erth_ml['mag'].max()) +
 ↪2)]

# Count the bins
erth_mag = pd.cut(erth_ml['mag'], bins=bins, right=True).value_counts()
erth_mag
```

```
[ ]: (1, 2]     3105
(0, 1]     2207
(2, 3]      862
(-1, 0]     491
(3, 4]      122
(4, 5]        2
(5, 6]        1
Name: mag, dtype: int64
```

3. Using the faang.csv file, group by the ticker and resample to monthly frequency. Make the following aggregations:

- Mean of the opening price
- Maximum of the high price
- Minimum of the low price
- Mean of the closing price
- Sum of the volume traded

```
[ ]: fngdf.agg({
    'open': np.mean,
    'high': np.max,
    'low': np.min,
```

```
    'close': np.sum
})
```

[ ]:
```
open         687.148081
high        2050.500000
low          123.020000
close     861617.430600
dtype: float64
```

4. Build a crosstab with the earthquake data between the tsunami column and the magType column. Rather than showing the frequency count, show the maximum magnitude that was observed for each combination. Put the magType along the columns.

[ ]:
```
pd.crosstab(
    index = erthdf['tsunami'],
    columns = erthdf['magType'],
    values=erthdf['mag'],
    aggfunc='max'
)
```

[ ]:

magType	mb	mb_lg	md	mh	ml	ms_20	mw	mwb	mwr	mww
tsunami										
0	5.6	3.5	4.11	1.1	4.2	NaN	3.83	5.8	4.8	6.0
1	6.1	NaN	NaN	NaN	5.1	5.7	4.41	NaN	NaN	7.5

5. Calculate the rolling 60-day aggregations of OHLC data by ticker for the FAANG data. Use the same aggregations as exercise no. 3.

[ ]:
```
# check the data type of date
fngdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1255 entries, 0 to 1254
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ticker  1255 non-null   object
 1   date    1255 non-null   object
 2   open    1255 non-null   float64
 3   high    1255 non-null   float64
 4   low     1255 non-null   float64
 5   close   1255 non-null   float64
 6   volume  1255 non-null   int64
dtypes: float64(4), int64(1), object(2)
memory usage: 68.8+ KB
```

[ ]:
```
# change the datatype of date from 'object' to 'datetime'
fngdf['date'] = pd.to_datetime(fngdf['date'])
```

```
fngdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1255 entries, 0 to 1254
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ticker  1255 non-null   object
 1   date    1255 non-null   datetime64[ns]
 2   open    1255 non-null   float64
 3   high    1255 non-null   float64
 4   low     1255 non-null   float64
 5   close   1255 non-null   float64
 6   volume  1255 non-null   int64
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 68.8+ KB
```

[ ]:
```
fng_roll = fngdf.groupby('ticker').rolling(window=60, min_periods=None,␣
 ↪on='date').agg({
    'open': np.mean,
    'high': np.max,
    'low': np.min,
    'close': np.sum
})

fng_roll
```

[ ]:
```
                         open      high     low      close
ticker date
AAPL   2018-01-02         NaN       NaN     NaN        NaN
       2018-01-03         NaN       NaN     NaN        NaN
       2018-01-04         NaN       NaN     NaN        NaN
       2018-01-05         NaN       NaN     NaN        NaN
       2018-01-08         NaN       NaN     NaN        NaN
…                         …       …     …          …
NFLX   2018-12-24  306.018050  386.7999  233.68  18194.390
       2018-12-26  303.596050  386.7999  231.23  18073.930
       2018-12-27  301.500383  386.7999  231.23  17948.065
       2018-12-28  299.393050  380.9300  231.23  17827.005
       2018-12-31  297.420217  380.0000  231.23  17717.615

[1255 rows x 4 columns]
```

6. Create a pivot table of the FAANG data that compares the stocks. Put the ticker in the rows and show the averages of the OHLC and volume traded data.

[ ]:
```
fng_pvt = pd.pivot_table(
    fngdf,
    index='ticker',
    values=['open', 'high', 'low', 'close'],
    aggfunc='mean'
    )

fng_pvt
```

[ ]:

	close	high	low	open
ticker				
AAPL	186.986218	188.906858	185.135729	187.038674
AMZN	1641.726175	1662.839801	1619.840398	1644.072669
FB	171.510936	173.615298	169.303110	171.454424
GOOG	1113.225139	1125.777649	1101.001594	1113.554104
NFLX	319.290299	325.224583	313.187273	319.620533

7. Calculate the Z-scores for each numeric column of Netflix's data (ticker is NFLX) using apply().

[ ]:
```
from scipy import stats

fng_z = fngdf[fngdf['ticker'] == 'NFLX'][['open', 'high', 'low', 'close',␣
↪'volume']].apply(stats.zscore)
fng_z
```

[ ]:

	open	high	low	close	volume
753	-2.505749	-2.521050	-2.415042	-2.421473	-0.088937
754	-2.385047	-2.428022	-2.290360	-2.339951	-0.508620
755	-2.300860	-2.410885	-2.239081	-2.328071	-0.961204
756	-2.279559	-2.350294	-2.206487	-2.238767	-0.783894
757	-2.223367	-2.299699	-2.148042	-2.196572	-1.040606
…	…	…	…	…	…
999	-1.574618	-1.521399	-1.630448	-1.749435	-0.339680
1000	-1.738529	-1.442855	-1.680690	-1.344082	0.518073
1001	-1.410097	-1.420618	-1.498794	-1.305267	0.135138
1002	-1.251257	-1.291594	-1.299877	-1.294718	-0.085334
1003	-1.206222	-1.124597	-1.090706	-1.057529	0.360163

[251 rows x 5 columns]

8. Add event descriptions:

- Create a dataframe with the following three columns: ticker, date, and event. The columns should have the following values:
- ticker: 'FB'
- date: ['2018-07-25', '2018-03-19', '2018-03-20']
- event: ['Disappointing user growth announced after close.', 'Cambridge Analytica story', 'FTC

investigation']
- Set the index to ['date', 'ticker']
- Merge this data with the FAANG data using an outer join

[ ]:
```
# Create new data frame
new_df = pd.DataFrame({'ticker': 'FB',
                       'date': ['2018-07-25', '2018-03-19', '2018-03-20'],
                       'event': ['Disappointing user growth announced after␣
↪close.', 'Cambridge Analytica story', 'FTC investigation']
                       })

# Set the index to ['date', 'ticker']
new_df.set_index(['date', 'ticker'])

# Merge this data with the FAANG data using an outer join
new_df = new_df.merge(fngdf, how='outer')

new_df
```

[ ]:
```
     ticker        date                                            event  \
0        FB  2018-07-25  Disappointing user growth announced after close.
1        FB  2018-03-19                         Cambridge Analytica story
2        FB  2018-03-20                                 FTC investigation
3        FB  2018-01-02                                               NaN
4        FB  2018-01-03                                               NaN
…       …         …                                               …
1250   GOOG  2018-12-24                                               NaN
1251   GOOG  2018-12-26                                               NaN
1252   GOOG  2018-12-27                                               NaN
1253   GOOG  2018-12-28                                               NaN
1254   GOOG  2018-12-31                                               NaN

          open     high      low    close     volume
0      215.715   218.62   214.27   217.50   64592585
1      177.010   177.17   170.06   172.56   88140060
2      167.470   170.20   161.95   168.15  129851768
3      177.680   181.58   177.55   181.42   18151903
4      181.880   184.78   181.33   184.67   16886563
…         …       …       …       …       …
1250   973.900  1003.54   970.11   976.22    1590328
1251   989.010  1040.00   983.00  1039.46    2373270
1252  1017.150  1043.89   997.00  1043.88    2109777
1253  1049.620  1055.56  1033.10  1037.08    1413772
1254  1050.960  1052.70  1023.59  1035.61    1493722

[1255 rows x 8 columns]
```

9. Use the transform() method on the FAANG data to represent all the values in terms of

the first date in the data. To do so, divide all the values for each ticker by the values for the first date in the data for that ticker. This is referred to as an index, and the data for the first date is the base (https://ec.europa.eu/eurostat/statistics-explained/ index.php/ Beginners:Statisticalconcept-Indexandbaseyear). When data is in this format, we can easily see growth over time. Hint: transform() can take a function name.

[ ]:
```
fng_trns = new_df.groupby('ticker')[['open', 'high', 'low', 'close', 'volume']].
↪transform(lambda x: x / x.iloc[0])
fng_trns
```

[ ]:

	open	high	low	close	volume
0	1.000000	1.000000	1.000000	1.000000	1.000000
1	0.820573	0.810402	0.793672	0.793379	1.364554
2	0.776348	0.778520	0.755822	0.773103	2.010320
3	0.823679	0.830574	0.828627	0.834115	0.281021
4	0.843150	0.845211	0.846269	0.849057	0.261432
…	…	…	…	…	…
1250	0.928993	0.940578	0.928131	0.916638	1.285047
1251	0.943406	0.974750	0.940463	0.976019	1.917695
1252	0.970248	0.978396	0.953857	0.980169	1.704782
1253	1.001221	0.989334	0.988395	0.973784	1.142383
1254	1.002499	0.986653	0.979296	0.972404	1.206986

[1255 rows x 5 columns]