Python for Data Analytics

Pandas II

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Jan. 6 – 17, 2020

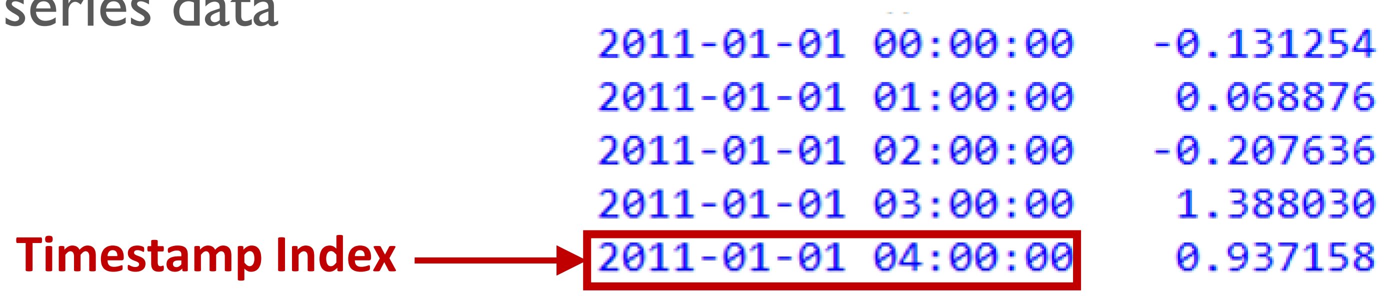
Outline

- Why Pandas?
- Pandas Series
- Pandas DataFrame
- I/O in Pandas
- Time Series Data in Pandas

Times Series Data in Pandas

Time Series Data

- How to analyze time series data?
  - Sample time series data



```
2011-01-01 00:00:00   -0.131254
2011-01-01 01:00:00    0.068876
2011-01-01 02:00:00   -0.207636
2011-01-01 03:00:00    1.388030
2011-01-01 04:00:00    0.937158
```

- Time series data is the data with the timestamp index
  - How to parse time series information from various sources and formats?
  - How to generate sequences of fixed-frequency dates and time spans
  - How to manipulate and convert date times with timezone information?
  - How to group data by time?
  - …

Python datetime Module

- `datetime.datetime` class: a combination of date and time
  - year, month, day, hour, minute, second, microsecond, tzinfo
- `datetime.now()`: return the current local datetime

```
>>> import datetime as dt
>>> now = dt.datetime.now()
>>> print(now)
2020-01-04 01:01:35.066589
>>> newyear = dt.datetime(2020, 1, 1)
>>> print(newyear)
2020-01-01 00:00:00
>>> print(now - newyear)
3 days, 1:01:35.066589
```

NumPy datetime64 Type

- NumPy supports datetime functionality with the data type called '`datetime64`'
  - No timezone support

```
>>> import numpy as np
>>> now = np.datetime64('now')
>>> print(now)
2020-01-03T16:39:44
>>> np.arange('2020-01', '2020-07', dtype='datetime64[M]')
array(['2020-01', '2020-02', '2020-03', '2020-04', '2020-05', '2020-06'],
      dtype='datetime64[M]')
>>> newyear = np.datetime64('2020-1-1')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: Error parsing datetime string "2020-1-1" at position 5
```

Converting to Datetime

- Pandas supports extensive capabilities and features for working with time series data based on NumPy `datetime64`.
- *pd*.`to_datetime(`*arg*, ...`)`
  - Convert argument to datetime.
  - Return type can be a DatetimeIndex, Series, or Timestamp
  - *arg*: integer, float, string, datetime, list, tuple, 1-D array, Series

```
>>> t = np.array(['1/8/2020', '2020/1/9', '20200110', '2020-1-13', '2020 1 14',
                  '2020, 1, 15', 'Jan. 16 2020', '17 Jan 2010'])
>>> pd.to_datetime(t)
DatetimeIndex(['2020-01-08', '2020-01-09', '2020-01-10', '2020-01-13',
               '2020-01-14', '2020-01-15', '2020-01-16', '2010-01-17'],
              dtype='datetime64[ns]', freq=None)
```

Generating DatetimeIndex

- *pd*.date_range([*start*], [*end*], [*periods*], [*freq*], ...)
  - Return a fixed frequency datetime index
  - *start*: Left bound for generating dates
  - *end*: Right bound for generating dates
  - *periods*: the number of datetime to generate
  - *freq*: the time interval between consecutive datetime values (default: 'D')

Freq string	Description	Freq string	Description
`'D'`	One absolute day	`'M'`	Calendar month end
`'H'`	One hour	`'MS'`	Calendar month begin
`'T' or 'min'`	One minute	`'BM'`	Business month end
`'S'`	One second	`'BMS'`	Business month begin
`'B'`	Business day (weekday)	`'WOM-2THU'`	Second Thursday of the month
`'W'`	One week	`'1h30min'`	One and half hour

date_range() Examples (1)

- Default: everyday

```
>>> pd.date_range('2020-1-6', '2020-1-17')
DatetimeIndex(['2020-01-06', '2020-01-07', '2020-01-08', '2020-01-09',
               '2020-01-10', '2020-01-11', '2020-01-12', '2020-01-13',
               '2020-01-14', '2020-01-15', '2020-01-16', '2020-01-17'],
              dtype='datetime64[ns]', freq='D')
```

- 14 days since 2020-1-6

```
>>> pd.date_range('2020 1 6', periods=14)
DatetimeIndex(['2020-01-06', '2020-01-07', '2020-01-08', '2020-01-09',
               '2020-01-10', '2020-01-11', '2020-01-12', '2020-01-13',
               '2020-01-14', '2020-01-15', '2020-01-16', '2020-01-17',
               '2020-01-18', '2020-01-19'],
              dtype='datetime64[ns]', freq='D')
```

date_range() Examples (2)

- Just weekdays

```
>>> pd.date_range('2020 1 1', '2020/1/20', freq='B')
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-06',
               '2020-01-07', '2020-01-08', '2020-01-09', '2020-01-10',
               '2020-01-13', '2020-01-14', '2020-01-15', '2020-01-16',
               '2020-01-17', '2020-01-20'],
              dtype='datetime64[ns]', freq='B')
```

- Every Sunday

```
>>> pd.date_range('2020-1-1', '2020-3-1', freq='W-SUN')
DatetimeIndex(['2020-01-05', '2020-01-12', '2020-01-19', '2020-01-26',
               '2020-02-02', '2020-02-09', '2020-02-16', '2020-02-23',
               '2020-03-01'],
              dtype='datetime64[ns]', freq='W-SUN')
```

date_range() Examples (3)

- First business day every two months

```
>>> pd.date_range('2020-1-1', '2020-12-31', freq='2BMS')
DatetimeIndex(['2020-01-01', '2020-03-02', '2020-05-01', '2020-07-01',
               '2020-09-01', '2020-11-02'],
              dtype='datetime64[ns]', freq='2BMS')
```

- Every one and half hour

```
>>> pd.date_range('2020-1-6 9:30', periods=7, freq='1h30min')
DatetimeIndex(['2020-01-06 09:30:00', '2020-01-06 11:00:00',
               '2020-01-06 12:30:00', '2020-01-06 14:00:00',
               '2020-01-06 15:30:00', '2020-01-06 17:00:00',
               '2020-01-06 18:30:00'],
              dtype='datetime64[ns]', freq='90T')
```

Finding the Day of the Week

- *pd*.DatetimeIndex.day_name(*self*, ...)
  - Return the day names of the DateTimeIndex

```
>>> idx = pd.date_range(start='2020-01-01', freq='D', periods=3)
>>> idx
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03'], dtype='datetime64[ns]', freq='D')

>>> idx.day_name()
Index(['Wednesday', 'Thursday', 'Friday'], dtype='object')

>>> war = pd.date_range(start='1950-6-25', freq='D', periods=3)
>>> war.day_name()
Index(['Sunday', 'Monday', 'Tuesday'], dtype='object')
```

Creating Time Series Data

- Create a range of `DatetimeIndex` object

```
>>> ts = pd.date_range('1/1/2020', periods=4, freq='2H')
>>> ts
DatetimeIndex(['2020-01-01 00:00:00', '2020-01-01 02:00:00',
               '2020-01-01 04:00:00', '2020-01-01 06:00:00'],
              dtype='datetime64[ns]', freq='2H')
```

- Use the `DatetimeIndex` object for Pandas Series or DataFrame index

```
>>> s=pd.Series(np.random.randn(len(ts)), index=ts)
>>> s
2020-01-01 00:00:00    0.872481
2020-01-01 02:00:00   -0.258136
2020-01-01 04:00:00    1.151232
2020-01-01 06:00:00    1.787104
Freq: 2H, dtype: float64
```

Example: Pandas Time Series Data (1)

- Create a dataframe: input dataset = <*timestamp, access count*>

```
import datetime as dt
import pandas as pd
data = {'date': ['2020-01-01 08:47:05.069722',
                 '2020-01-01 18:47:05.119994',
                 '2020-01-02 08:47:05.178768',
                 '2020-01-02 13:47:05.230071',
                 '2020-01-02 18:47:05.230071',
                 '2020-01-02 23:47:05.280592',
                 '2020-01-03 08:47:05.332662',
                 '2020-01-03 18:47:05.385109',
                 '2020-01-04 08:47:05.436523',
                 '2020-01-04 18:47:05.486877'],
 'counts': [34, 25, 26, 15, 15, 14, 26, 25, 62, 41]}
df = pd.DataFrame(data)
df
```

	date	counts
0	2020-01-01 08:47:05.069722	34
1	2020-01-01 18:47:05.119994	25
2	2020-01-02 08:47:05.178768	26
3	2020-01-02 13:47:05.230071	15
4	2020-01-02 18:47:05.230071	15
5	2020-01-02 23:47:05.280592	14
6	2020-01-03 08:47:05.332662	26
7	2020-01-03 18:47:05.385109	25
8	2020-01-04 08:47:05.436523	62
9	2020-01-04 18:47:05.486877	41

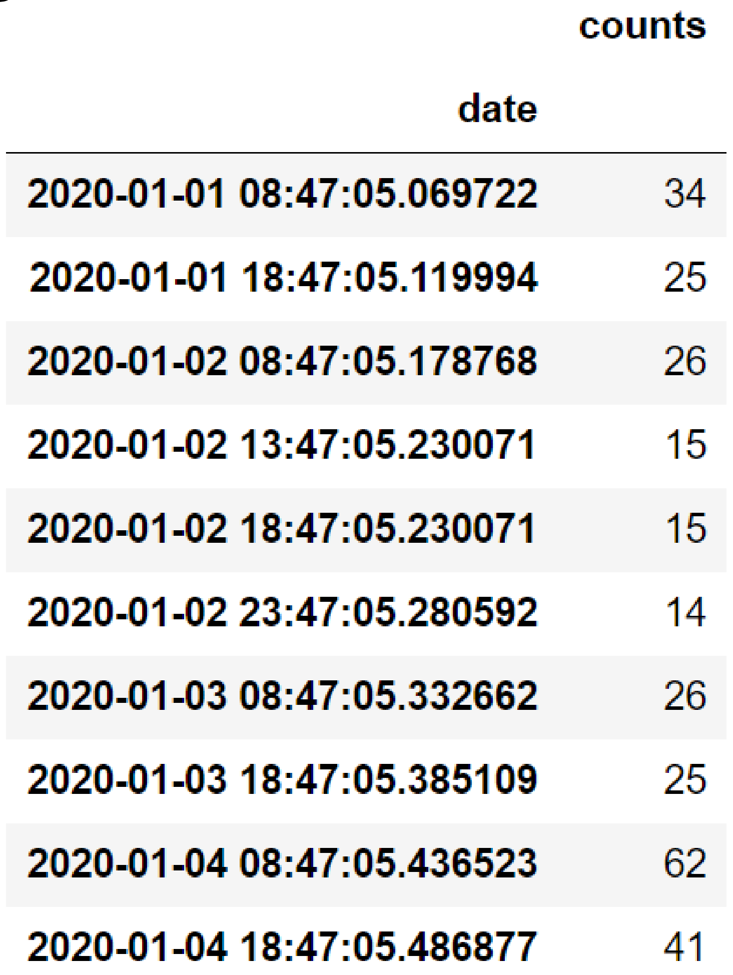
Example: Pandas Time Series Data (2)

- Convert `df['date']` from string to datetime

```
df['date'] = pd.to_datetime(df.date)
```

- Set `df['date']` as the index

```
df = df.set_index('date')
df
```



date	counts
2020-01-01 08:47:05.069722	34
2020-01-01 18:47:05.119994	25
2020-01-02 08:47:05.178768	26
2020-01-02 13:47:05.230071	15
2020-01-02 18:47:05.230071	15
2020-01-02 23:47:05.280592	14
2020-01-03 08:47:05.332662	26
2020-01-03 18:47:05.385109	25
2020-01-04 08:47:05.436523	62
2020-01-04 18:47:05.486877	41

Example: Pandas Time Series Data (3)

- View data in 2020

```
df['2020']
```

date	counts
2020-01-01 08:47:05.069722	34
2020-01-01 18:47:05.119994	25
2020-01-02 08:47:05.178768	26
2020-01-02 13:47:05.230071	15
2020-01-02 18:47:05.230071	15
2020-01-02 23:47:05.280592	14
2020-01-03 08:47:05.332662	26
2020-01-03 18:47:05.385109	25
2020-01-04 08:47:05.436523	62
2020-01-04 18:47:05.486877	41

- View data in January 2020

```
df['2020-01']
```

date	counts
2020-01-01 08:47:05.069722	34
2020-01-01 18:47:05.119994	25
2020-01-02 08:47:05.178768	26
2020-01-02 13:47:05.230071	15
2020-01-02 18:47:05.230071	15
2020-01-02 23:47:05.280592	14
2020-01-03 08:47:05.332662	26
2020-01-03 18:47:05.385109	25
2020-01-04 08:47:05.436523	62
2020-01-04 18:47:05.486877	41

Example: Pandas Time Series Data (4)

- Observations after Jan. 3, 2020

```
df[dt.datetime(2020, 1, 3):]
```

	counts
date	
2020-01-03 08:47:05.332662	26
2020-01-03 18:47:05.385109	25
2020-01-04 08:47:05.436523	62
2020-01-04 18:47:05.486877	41

- Observations between Jan. 1 - 2

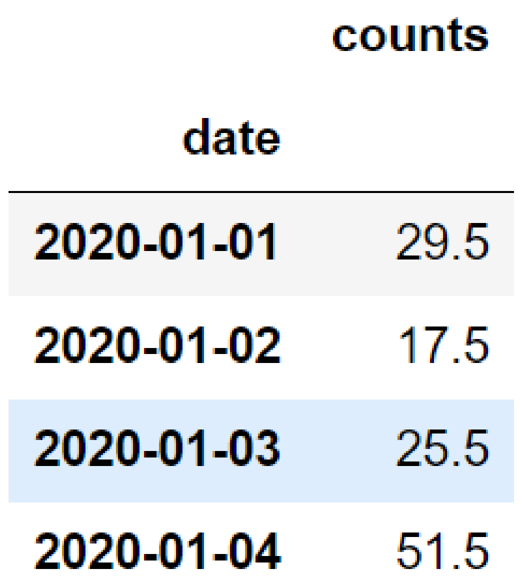
```
df['1/1/2020':'1/2/2020']
```

	counts
date	
2020-01-01 08:47:05.069722	34
2020-01-01 18:47:05.119994	25
2020-01-02 08:47:05.178768	26
2020-01-02 13:47:05.230071	15
2020-01-02 18:47:05.230071	15
2020-01-02 23:47:05.280592	14

Example: Pandas Time Series Data (5)

- Mean value of counts per day

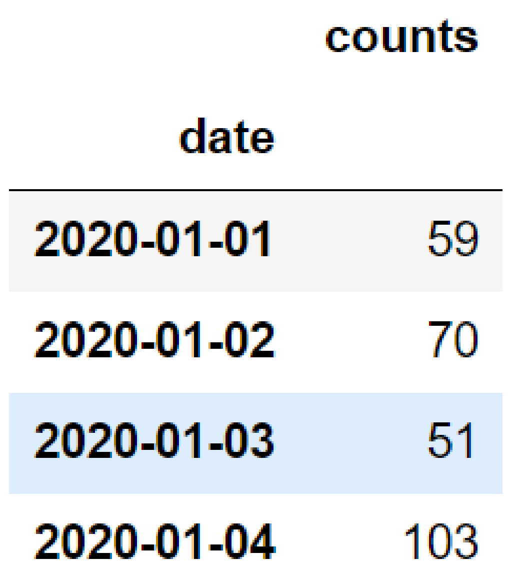
```
df.resample('D').mean()
```



	counts
date	
2020-01-01	29.5
2020-01-02	17.5
2020-01-03	25.5
2020-01-04	51.5

- Total value of counts per day

```
df.resample('D').sum()
```



	counts
date	
2020-01-01	59
2020-01-02	70
2020-01-03	51
2020-01-04	103

Example: Pandas Time Series Data (6)

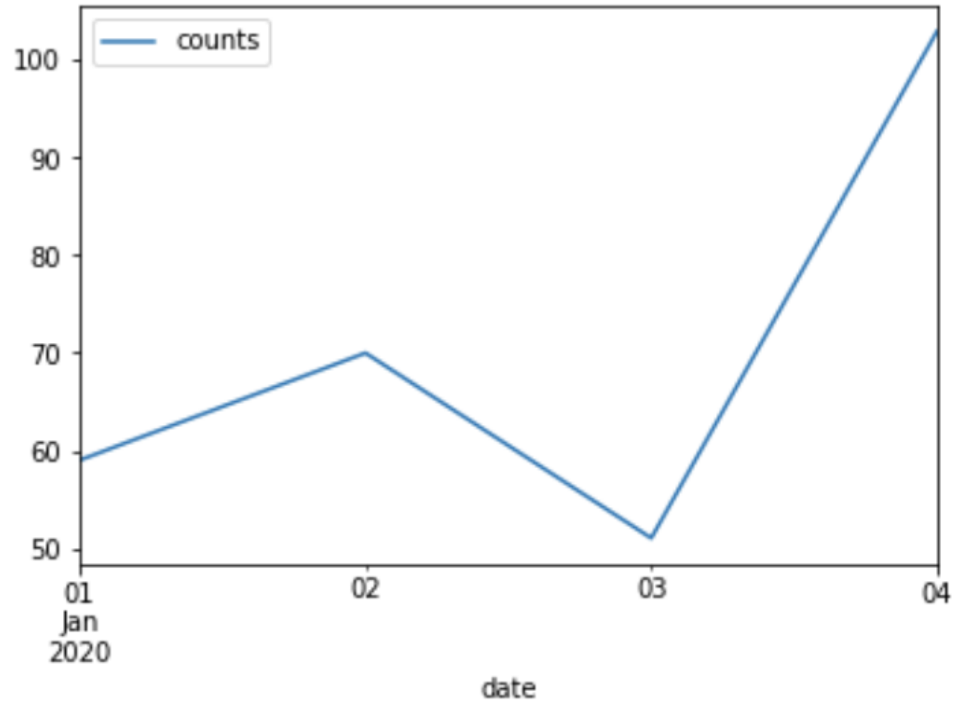
- Truncate observations after Jan. 3, 2020

```
df.truncate(after='1/3/2020')
```

	counts
date	
2020-01-01 08:47:05.069722	34
2020-01-01 18:47:05.119994	25
2020-01-02 08:47:05.178768	26
2020-01-02 13:47:05.230071	15
2020-01-02 18:47:05.230071	15
2020-01-02 23:47:05.280592	14

- Plot the total counts per day

```
import matplotlib.pyplot as plt
df.resample('D').sum().plot()
plt.show()
```


counts
Jan
date