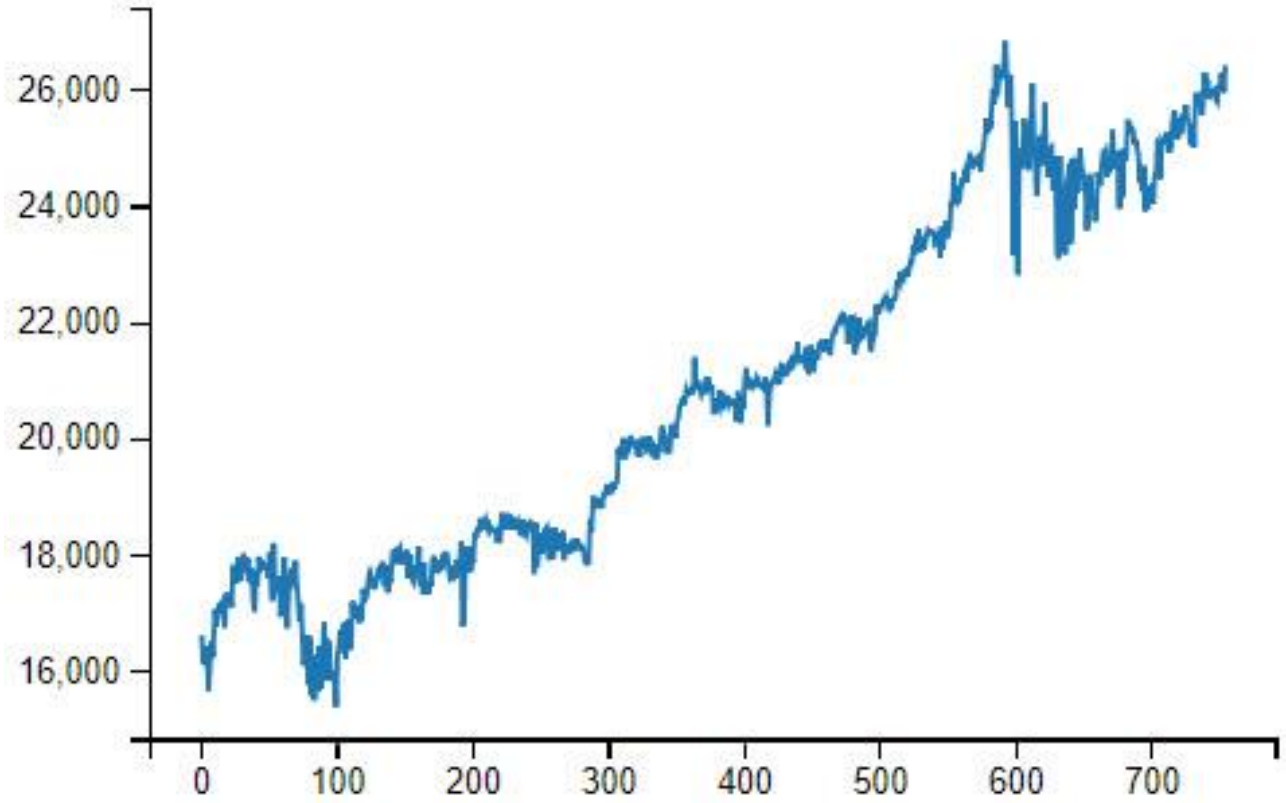
Time Series

Data handling

Director of TEAMLAB
Sungchul Choi

Time Series Data



우리가 만지는 대부분의 데이터는 시계열 데이터

시간에 특화된 기능이 필요함

Time series in Pandas

- 시간에 특화된 Groupby 기능이 필요함

  예: 데이터중 주말 통계만 필요함

- Time lag 또는 Moving Average는 어떻게 계산?

Pandas에는 이에 특화된 기능을 제공함

DateTimeIndex

Python의 Datetime 모듈

- 파이썬은 날짜 데이터 처리를 위해 datetime 모듈을 활용

```
from datetime import datetime
date_str = '09-19-2018'

date_object = datetime.strptime(date_str, '%m-%d-%Y').date()
print(type(date_object))
print(date_object)  # printed in default formatting
```

Python의 Datetime 모듈

```
date_object.day
```

```
19
```

```
date_object.month
```

```
9
```

```
date_object.weekday()
```

```
2
```

DateTime Index 만들기

- 대부분의 데이터는 str 으로 되어 있음

  → 호출 후 DateTime index로 변환이 필요함

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

DateTime Index 만들기

gwangju_eco.txt

파일을 호출해서 Datetime 인덱스를 만들어보세요!

유의사항 – 1) CP949 Encoding 2) Crosstab

3) separation 4) Transpose

	0	1
시도별	시도별	광주광역시
2016. 01	경제활동인구 (천명)	746.1
2016. 01.1	취업자 (천명)	722.3
2016. 01.2	실업자 (천명)	23.8
2016. 01.3	실업률 (%)	3.2
2016. 01.4	고용률 (%)	57.5
2016. 02	경제활동인구 (천명)	752.7
2016. 02.1	취업자 (천명)	722.4
2016. 02.2	실업자 (천명)	30.3
2016. 02.3	실업률 (%)	4
2016. 02.4	고용률 (%)	57.5
2016. 03	경제활동인구 (천명)	738.8

Hint 코드

```
df["date"].str.split(".", expand=True)
```

Concat후 Merge

date index를 timeindex로

Type 변경 – str → value

category date	경제활동인구 (천명)	고용률 (%)	실업률 (%)	실업자 (천명)	취업자 (천명)
2016-01-01	746.1	57.5	3.2	23.8	722.3
2016-02-01	752.7	57.5	4.0	30.3	722.4
2016-03-01	738.8	57.2	2.8	20.7	718.2
2016-04-01	748.1	57.8	2.9	21.5	726.6
2016-05-01	758.1	58.7	2.7	20.4	737.7
2016-06-01	763.4	58.8	3.1	23.6	739.8
2016-07-01	760.1	58.7	2.8	21.6	738.5
2016-08-01	758.9	58.3	3.4	25.6	733.4
2016-09-01	755.8	57.9	3.6	26.9	728.9
2016-10-01	755.3	58.2	3.0	22.7	732.6
2016-11-01	747.7	57.9	2.6	19.6	728.1
2016-12-01	752.6	58.1	2.8	21.3	731.3
2017-01-01	747.1	58.2	1.9	13.9	733.2
2017-02-01	756.0	58.0	3.4	25.6	730.4
2017-03-01	757.8	58.6	2.7	20.2	737.6
2017-04-01	769.0	58.8	3.7	28.3	740.7
2017-05-01	772.6	59.2	3.4	26.4	746.2

Kaggle Challenge


kaggle™

Data analysis Competition

Google is acquiring data science community Kaggle

Posted Mar 7, 2017 by Frederic Lardinois (@fredericl), Matthew Lynley (@mattlynley), John Mannes (@JohnMannes)

Crunchbase

Kaggle –

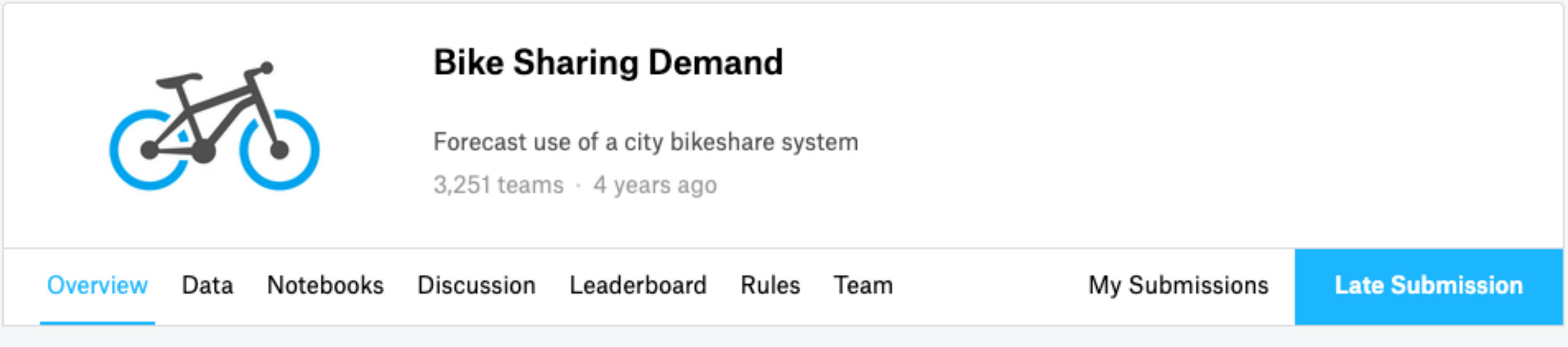
FOUNDED
2010

OVERVIEW

기업은 데이터를
분석가는 분석기법을

Bike Demand

고전적인 시계열 데이터 문제 – 시각화와 전처리에 용이함



Data 불러오기

```
df = pd.read_csv(
        os.path.join(DATA_DIR, "train.csv"),
    parse_dates = ['datetime'])
df.set_index("datetime", inplace=True)
df.head()
```

Resampling

Time resampling

- 분석 목표에 따라 시간 기준으로 데이터를 Aggregation

- Groupby와 유사 → 훨씬 간단하고 다양한 기능 제공

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

월별 자건거 수요량을 보고 싶다?

GroupBy

```
df["month"] = df.index.month
df["year"] = df.index.year

df.groupby(
    ["year", "month"])["count"].sum().reset_index()
```

Resampling

```
df["count"].resample('Q').sum()
df["count"].resample('M').sum()
df["count"].resample('D').sum()
df["count"].resample('W').sum()
```

Resampling

Frequnecy 기준을 뽑는

알파벳 제공

https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html

Alias	Description
B	business day frequency
C	custom business day frequency (experimental)
D	calendar day frequency
W	weekly frequency
M	month end frequency
BM	business month end frequency
CBM	custom business month end frequency
MS	month start frequency
BMS	business month start frequency
CBMS	custom business month start frequency
Q	quarter end frequency
BQ	business quarter endfrequency
QS	quarter start frequency
BQS	business quarter start frequency
A	year end frequency
BA	business year end frequency
AS	year start frequency
BAS	business year start frequency
H	hourly frequency
T	minutely frequency
S	secondly frequency
L	milliseonds
U	microseconds
N	nanoseconds

Resampling - Filter

- something_range 함수로 기간을 생성하여 Filter 지정

```
period = pd.date_range(
    start='2011-01-01', end='2011-05-31', freq='M')
df["count"].resample('M').sum()[period]
```

pandas.date_range

pandas.**date_range**(*start=None*, *end=None*, *periods=None*, *freq=None*, *tz=None*, *normalize=False*, *name=None*, *closed=None*, ***kwargs*) [source]

Return a fixed frequency DatetimeIndex.

start : *str or datetime-like, optional*

Left bound for generating dates.

end : *str or datetime-like, optional*

Right bound for generating dates.

periods : *integer, optional*

Number of periods to generate.

freq : *str or DateOffset, default 'D'*

Frequency strings can have multiples, e.g. '5H'. See here for a list of frequency aliases.

```
>>> pd.date_range(start='1/1/2018', end='1/08/2018')
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04',
               '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08'],
              dtype='datetime64[ns]', freq='D')
```

Specify *start* and *periods*, the number of periods (days).

```
>>> pd.date_range(start='1/1/2018', periods=8)
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04',
               '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08'],
              dtype='datetime64[ns]', freq='D')
```

Specify *end* and *periods*, the number of periods (days).

```
>>> pd.date_range(end='1/1/2018', periods=8)
DatetimeIndex(['2017-12-25', '2017-12-26', '2017-12-27', '2017-12-28',
               '2017-12-29', '2017-12-30', '2017-12-31', '2018-01-01'],
              dtype='datetime64[ns]', freq='D')
```

See also:

`DatetimeIndex`
: An immutable container for datetimes.

`timedelta_range`
: Return a fixed frequency TimedeltaIndex.

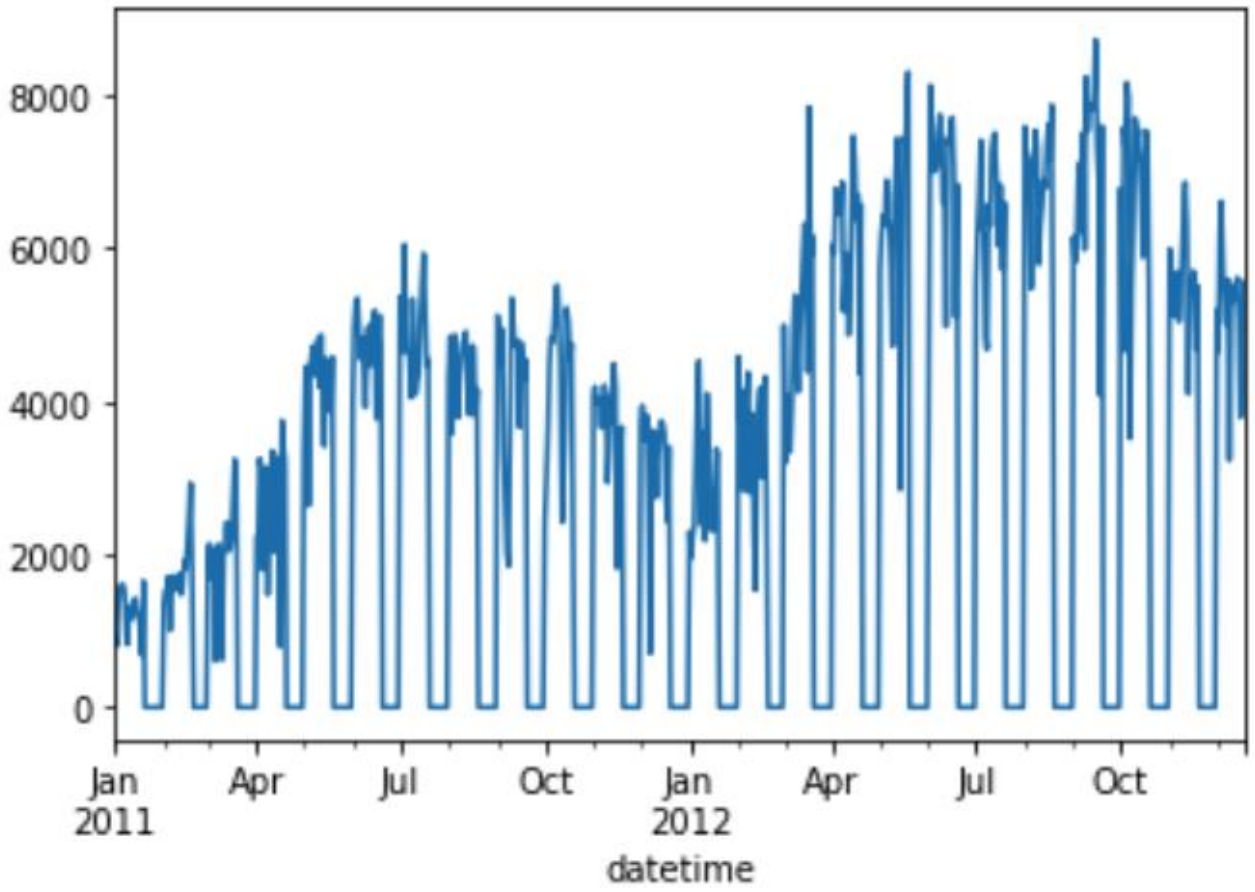
`period_range`
: Return a fixed frequency PeriodIndex.

`interval_range`
: Return a fixed frequency IntervalIndex.

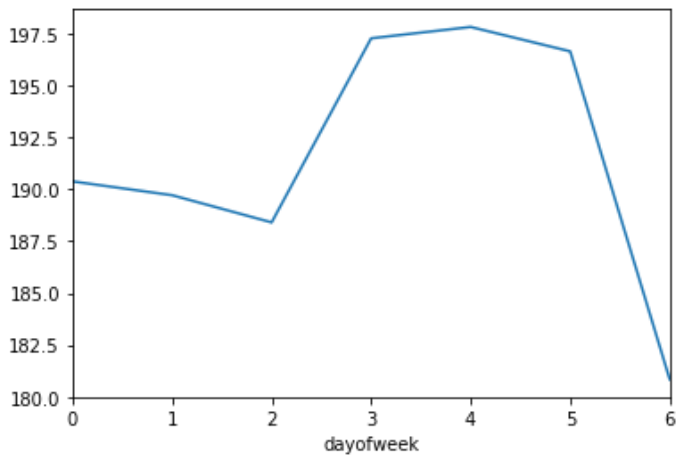
Resampling - chart

```
df["count"].resample('D').sum().plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1301cfeb8>
```



요일별 자전거 수요량의 평균을 보고 싶다면?




```
df["dayofweek"] = df.index.dayofweek
df.groupby("dayofweek")["count"].mean()
    kind="line")
```

Time shifting

Time shifting

- 시간의 차(Time Lag) 분석 필요

  예) 30일전에 비해 주가는 상승세인가?

- Pandas내 Time shifting 기능으로

  time window를 기준으로 기간의 차이를 분석가능

Time shifting

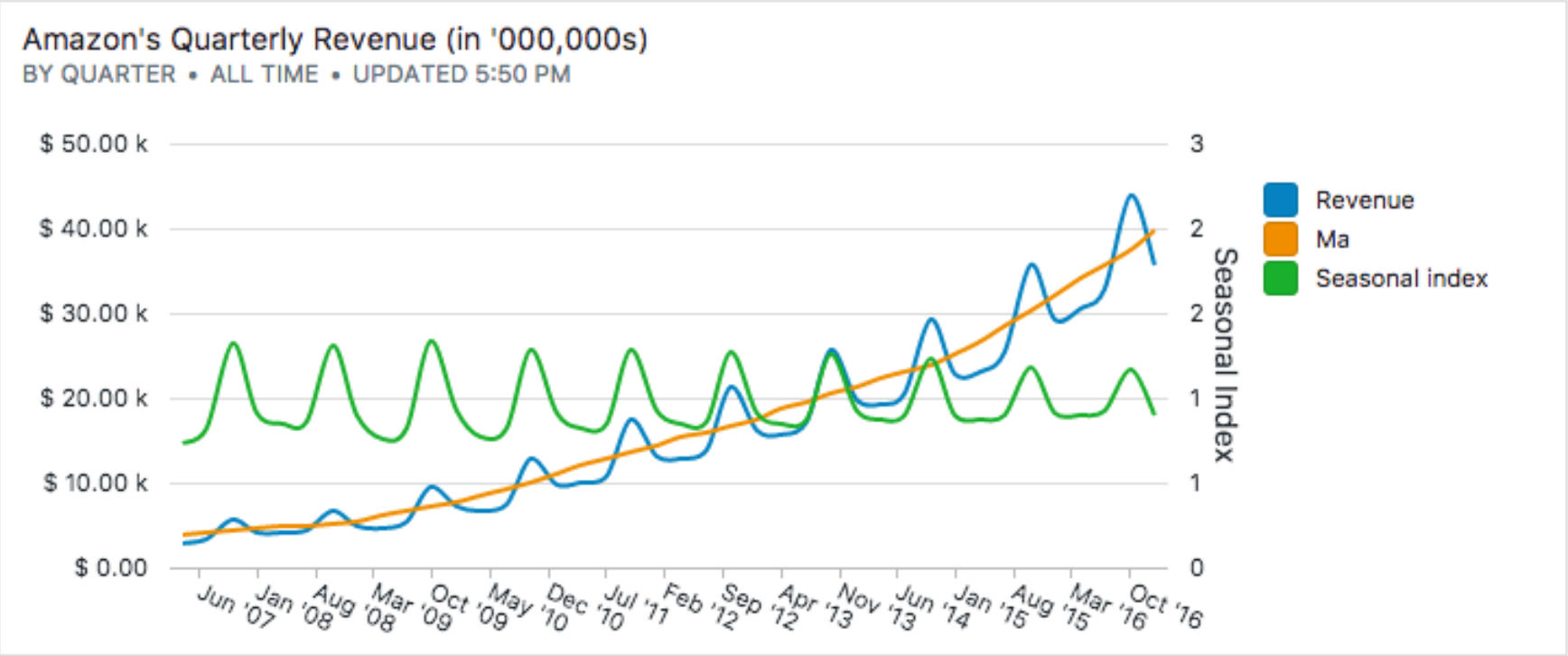
```
monthly_mean = df["count"].resample('M').mean()
monthly_mean_shift = monthly_mean.shift(
                    periods=2, fill_value=0)
df_monthly = pd.DataFrame(
          monthly_mean, columns=["count"])
df_monthly["2_shift_demand"] \
              = monthly_mean_shift
df_monthly
```

Moving average

Moving average

- 시계열 데이터는 노이즈 발생

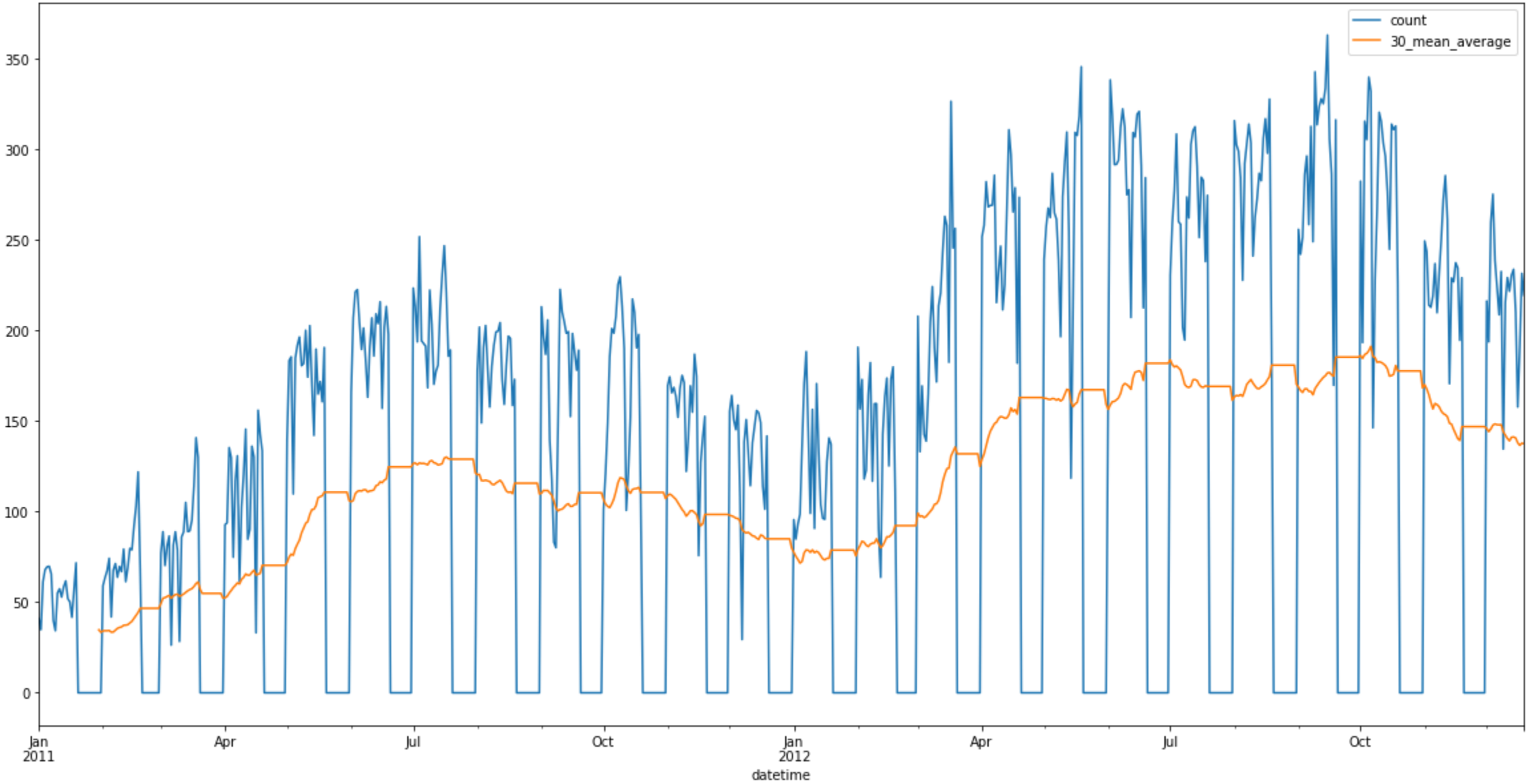
  → 노이즈를 줄이면서 추세를 보기 위한 이동평균법



Moving average

```
monthly_mean = df["count"].resample(
        'D').mean().fillna(0)
monthly_mean_shift = monthly_mean.rolling(
        window=30, ).mean()
df_monthly = pd.DataFrame(
    monthly_mean, columns=["count"])
df_monthly["30_mean_average"] = monthly_mean_shift
df_monthly.plot(figsize=(20,10))
```

Moving average



Cumsum

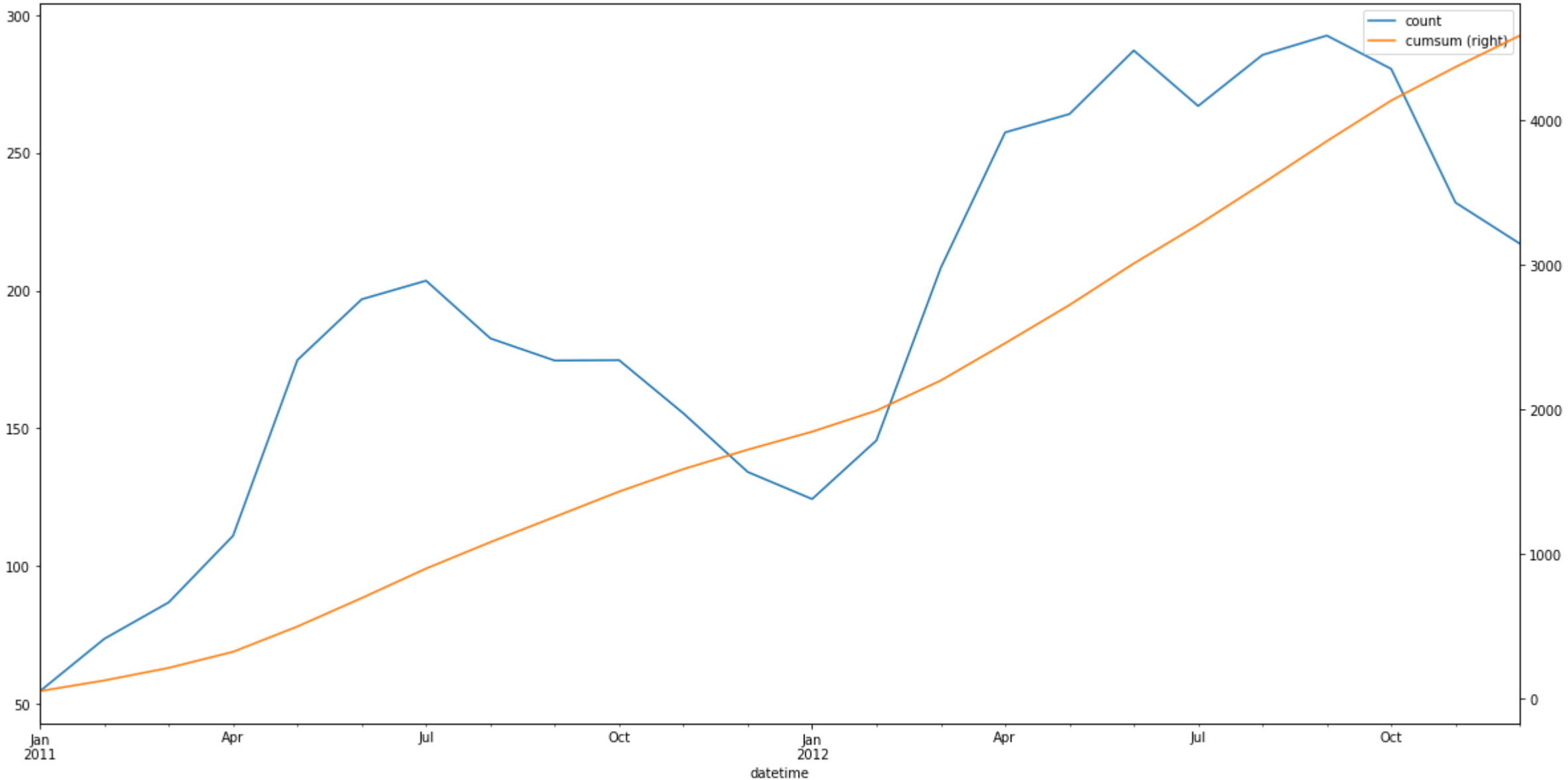
- 시계열 데이터를 window 마다 합침
- rolling(window=10).sum() 과 다름

```
monthly_mean = df["count"].resample('M').mean()
cumsum = monthly_mean.cumsum()
```

Secondary axis

```
ax = df_monthly.plot(y='count', use_index=True)
df_monthly.plot(
    y='cumsum',secondary_y=True,
    ax=ax, use_index=True, figsize=(20,10))
```

Secondary axis



Human knowledge belongs to the world.