Lesson 12 Summary: Time Series

Data Science with Python – Key Concepts

Data Science Program

Time Series Overview

Time Series Operations

Parse	Resample	Shift	Rolling
`pd.to_datetime()`	`df.resample()`	`df.shift()`	`df.rolling()`
Convert dates	Change freq	Lag/lead	Windows

Frequency Codes:

`D (day) | W (week) | M (month) | Q (quarter) | Y (year)`

DatetimeIndex enables powerful date-based operations

Time series operations are fundamental to financial analysis

DatetimeIndex

Working with dates in pandas:

- **Parse:** `pd.to_datetime('2024-01-15')`
- **Range:** `pd.date_range('2024-01-01', periods=100)`
- **Set index:** `df.set_index('Date')`

Date slicing:

```
df['2024'] # All of 2024
df['2024-01':'2024-03'] # Q1 2024
```

DatetimeIndex enables powerful date-based slicing

Resampling

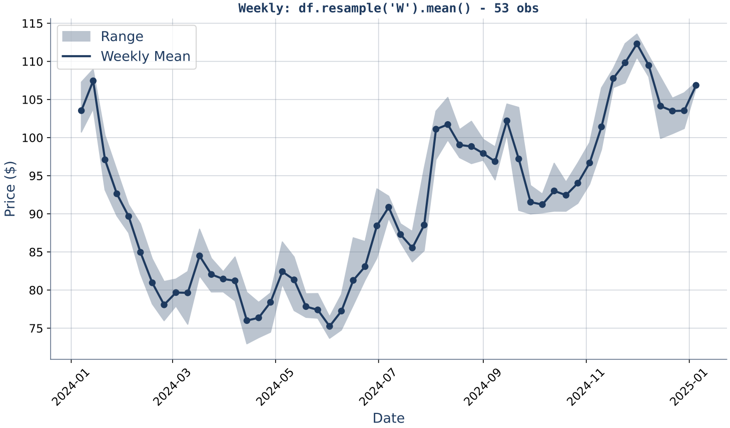
Change data frequency:

- **Downsample:** `df.resample('M').mean()` – Daily to monthly
- **OHLC:** `df.resample('W').ohlc()` – Open/High/Low/Close
- **Upsample:** `df.resample('H').ffill()` – Fill higher freq

Frequency codes:
D=day, W=week, M=month, Q=quarter, Y=year, B=business day

resample() is like groupby for time series

Weekly Resampling



Resample daily data to weekly for trend analysis

Shifting and Lagging

Move data forward or backward in time:

- **Lag:** `df['Price'].shift(1)` – Previous value
- **Lead:** `df['Price'].shift(-1)` – Next value
- **Returns:** `df['Price'].pct_change()`

Calculate returns:

```
df['Return'] = df['Price'] / df['Price'].shift(1) - 1
df['LogReturn'] = np.log(df['Price'] / df['Price'].shift(1))
```

shift() is essential for return calculations

Rolling Windows

Calculate statistics over moving windows:

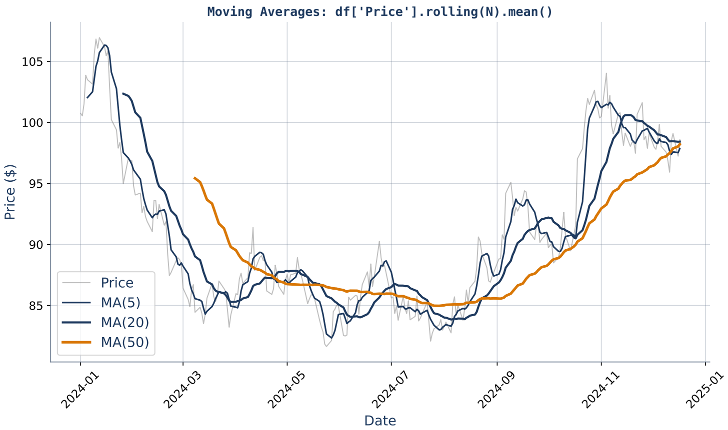
- **Moving average:** `df['Price'].rolling(20).mean()`
- **Volatility:** `df['Return'].rolling(20).std()`
- **Min/Max:** `df['Price'].rolling(20).max()`

Bollinger Bands:

```
ma = df['Price'].rolling(20).mean()
std = df['Price'].rolling(20).std()
upper = ma + 2*std
lower = ma - 2*std
```

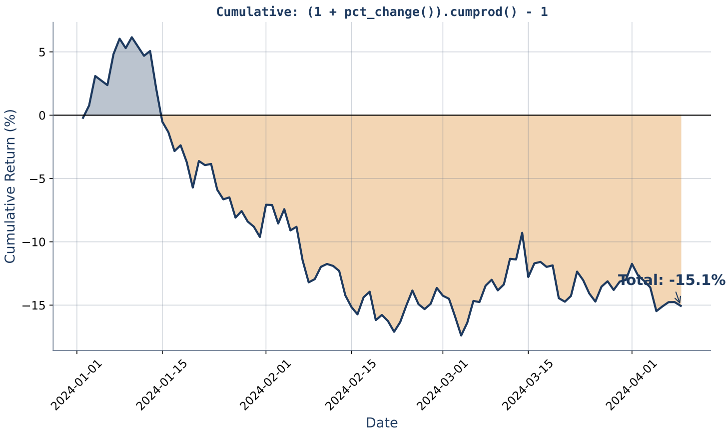
rolling() enables trend and volatility analysis

Moving Averages



Compare short vs long moving averages for signals

Cumulative Returns



cumsum()/cumprod() track performance over time

Quick Reference

Essential Time Series Operations:

Operation	Syntax
Parse dates	`pd.to_datetime(col)`
Date range	`pd.date_range(start, periods=n)`
Resample	`df.resample('M').mean()`
Shift/lag	`df['col'].shift(1)`
Pct change	`df['col'].pct_change()`
Rolling mean	`df['col'].rolling(n).mean()`
Cumulative	`(1 + returns).cumprod()`
Date slice	`df['2024-01':'2024-06']`

Time series tools are essential for financial analysis