Pandas: Séries Temporais

Datas, Indexação e Re-amostragem para Engenharia Agrícola

Curso Básico de Ciência de Dados para Engenharia Agrícola

O que são Séries Temporais?

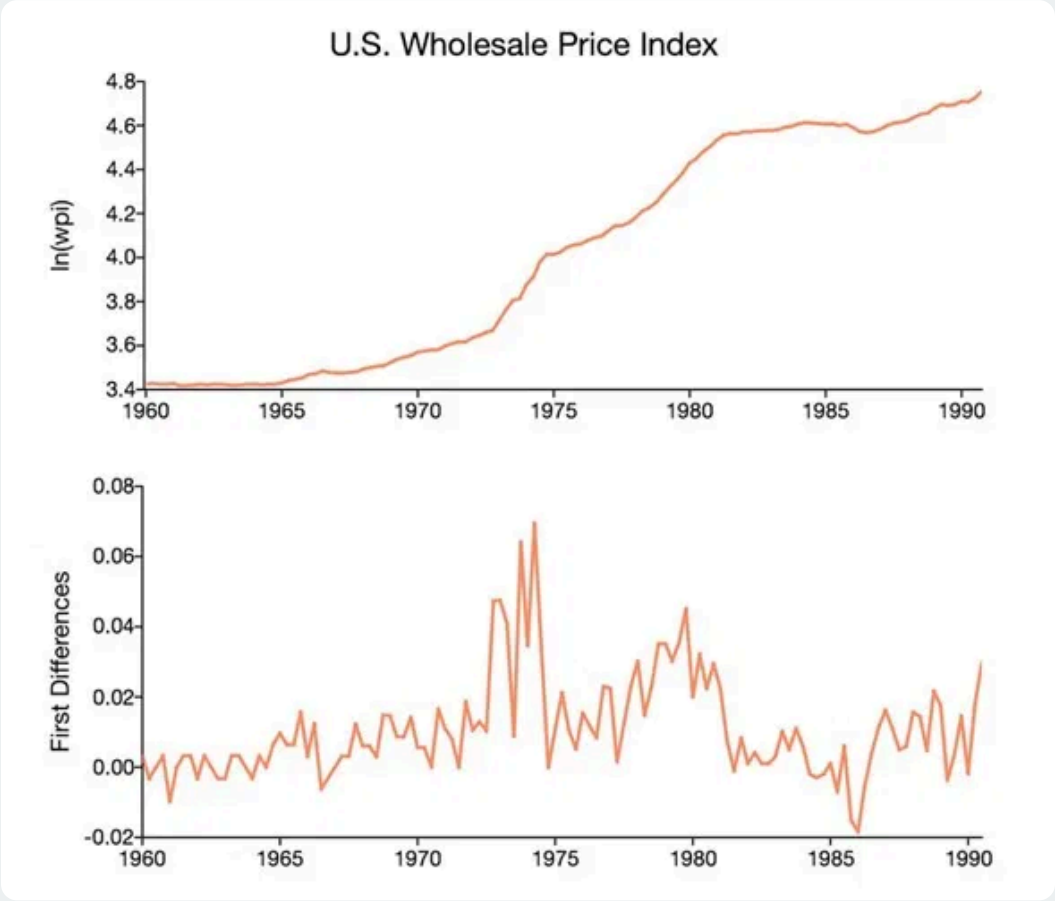
Séries temporais são dados coletados ou indexados em intervalos de tempo sucessivos, onde a ordem temporal é fundamental para a análise.

Características principais:

- Dependência temporal entre observações
- Presença de sazonalidade (padrões cíclicos)
- Tendências de longo prazo
- Variações irregulares ou aleatórias

Importância na Engenharia Agrícola:

- Monitoramento de culturas ao longo do tempo
- Previsão de safras e produtividade
- Gestão de recursos hídricos e irrigação
- Análise de dados climáticos e meteorológicos
- Detecção de padrões sazonais em pragas e doenças



Pandas e Séries Temporais

Por que Pandas?

O Pandas é uma biblioteca Python poderosa e flexível para análise e manipulação de dados, especialmente adequada para séries temporais.

- Estruturas de dados otimizadas para séries temporais
- Funções específicas para manipulação de datas e horas
- Ferramentas robustas para indexação temporal
- Métodos eficientes para re-amostragem de dados
- Integração com outras bibliotecas científicas (NumPy, Matplotlib)

Objetivos da Aula

- Compreender os tipos de dados temporais no Pandas
- Aprender a converter e criar séries de datas
- Dominar a indexação de dados por tempo
- Entender o conceito de re-amostragem
- Aplicar downsampling e upsampling em dados agrícolas
- Explorar aplicações práticas na Engenharia Agrícola

Ao final desta aula, você será capaz de manipular e analisar dados temporais agrícolas de forma eficiente usando Pandas!

Datas e Horas no Pandas: Tipos de Dados

Timestamp

Representa um único ponto no tempo (data e hora específica). Equivalente ao `datetime.datetime` da biblioteca padrão Python.

```
# Criando um Timestamp
ts = pd.Timestamp('2023-09-23 10:30:00')

# Acessando componentes
print(ts.year) # 2023
print(ts.month) # 9
print(ts.day) # 23
```

DatetimeIndex

Uma sequência de `Timestamps`, ideal para indexar `Series` e `DataFrames`. Permite operações eficientes de seleção e filtragem baseadas em tempo.

```
# Criando um DatetimeIndex
dti =
pd.DatetimeIndex(['2023-01-01',
'2023-01-02',
'2023-01-03'])

# Usando como índice
s = pd.Series([10, 20, 30],
index=dti)
```

Timedelta

Representa uma duração ou diferença entre dois `Timestamps`. Útil para cálculos de intervalos de tempo, como dias entre plantio e colheita.

```
# Criando um Timedelta
td = pd.Timedelta('1 day')

# Operações com datas
data_inicial =
pd.Timestamp('2023-05-01')
data_final = data_inicial +
td # 2023-05-02
```

Conversão de Dados para Datas

pd.to_datetime()

A função **pd.to_datetime()** é fundamental para converter strings e outros formatos para o tipo **datetime64[ns]** do Pandas.

Benefícios:

- Reconhece automaticamente diversos formatos de data
- Permite operações aritméticas com datas
- Habilita indexação e seleção baseada em tempo
- Possibilita o uso de métodos específicos para séries temporais

> **Dica:** Sempre converta colunas de data para o tipo datetime64 antes de realizar análises temporais. Isso garante acesso a todos os recursos de séries temporais do Pandas.

Exemplo Prático

```
import pandas as pd

# Dados de exemplo (temperatura diária em uma fazenda)
data = {'Data': ['2023-01-01', '2023-01-02', '2023-01-03',
          '2023-01-04', '2023-01-05'],
        'Temperatura_C': [25.1, 26.5, 24.9, 27.0, 25.5]}
df = pd.DataFrame(data)

# Converter a coluna 'Data' para o tipo datetime
df['Data'] = pd.to_datetime(df['Data'])
```

Resultado:

```
        Data  Temperatura_C
0 2023-01-01           25.1
1 2023-01-02           26.5
2 2023-01-03           24.9
3 2023-01-04           27.0
4 2023-01-05           25.5

Tipos de dados:
Data             datetime64[ns]
Temperatura_C           float64
dtype: object
```

Formatos aceitos: '2023-01-01', '01/01/2023', '01-Jan-2023', 'Jan 1, 2023', timestamps, objetos datetime, e muitos outros.

Criação de Séries de Datas

pd.date_range()

A função pd.date_range() gera um DatetimeIndex com uma frequência específica, ideal para criar índices temporais para séries e dataframes.

Parâmetros Principais:

- **start**: Data inicial
- **end**: Data final (opcional)
- **periods**: Número de períodos (opcional)
- **freq**: Frequência ('D' para diário, 'W' para semanal, 'M' para mensal, 'h' para horário, etc.)

```
import pandas as pd

# Gerar um índice de datas diárias para um mês
datas_diarias = pd.date_range(start="2023-03-01",
                              periods=31,
                              freq="D")
print("Datas diárias:")
print(datas_diarias[:5])  # Mostrando apenas as 5 primeiras

# Gerar um índice de datas semanais
datas_semanais = pd.date_range(start="2023-01-01"
```

Aplicações

Criar séries temporais com índices de datas é fundamental para análise de dados agrícolas, como registros de temperatura, precipitação ou crescimento de culturas.

```
# Criar uma Série com índice de datas
temperaturas = pd.Series(
    [20, 22, 21, 23, 24],  # Valores de temperatura
    index=pd.date_range(start="2023-04-01",
                        periods=5,
                        freq="D")  # Índice de datas
)
print("\nSérie com índice de datas:")
print(temperaturas)
```

```
Série com índice de datas:
2023-04-01    20
2023-04-02    22
2023-04-03    21
2023-04-04    23
2023-04-05    24
Freq: D, dtype: int64
```

Dica: Frequências comuns em dados agrícolas:

D: Diário (registros meteorológicos)
W: Semanal (monitoramento de culturas)
M: Mensal (análises de safra)
h: Horário (dados de sensores de irrigação)

Indexação de Séries Temporais

Importância da Indexação Temporal

A indexação temporal permite selecionar e filtrar dados de forma eficiente usando o tempo como referência, facilitando análises específicas por períodos.

Métodos de Indexação

- Por data exata: df.loc['YYYY-MM-DD']
- Por período (fatiamento): df.loc['início':'fim']
- Por ano/mês: df.loc['YYYY'], df.loc['YYYY-MM']
- Por strings parciais: Permite selecionar usando apenas parte da data

Aplicações na Engenharia Agrícola:

- Análise de dados climáticos por estação
- Monitoramento de umidade do solo durante períodos críticos
- Comparação de produtividade entre safras

Exemplo Prático

```
import pandas as pd
import numpy as np

# Criar DataFrame com dados de umidade do solo
idx = pd.date_range(start="2023-01-01", periods=20, freq="6h")
umidade_solo = pd.Series(np.random.uniform(0.2, 0.6, size=20),
                         index=idx)
df_umidade = pd.DataFrame({"Umidade_Solo": umidade_solo})

# Selecionar dados de um dia específico
dia_especifico = df_umidade.loc['2023-01-02']

# Selecionar dados de um período
periodo = df_umidade.loc['2023-01-01 12:00':'2023-01-02 18:00']
```

Umidade do solo em 2023-01-02:

	Umidade_Solo
2023-01-02 00:00:00	0.339690
2023-01-02 06:00:00	0.456680
2023-01-02 12:00:00	0.594025
2023-01-02 18:00:00	0.325516

💡 **Dica:** Para dados agrícolas, a indexação temporal facilita a análise de ciclos de cultivo, períodos de irrigação e eventos climáticos específicos.

Re-amostragem (Resampling)

Re-amostragem é o processo de alterar a frequência das observações em uma série temporal, permitindo analisar os dados em diferentes escalas temporais.

↓ Downsampling

Reduz a frequência dos dados (ex: de horário para diário). Requer **agregação** dos valores (soma, média, máximo, etc.).

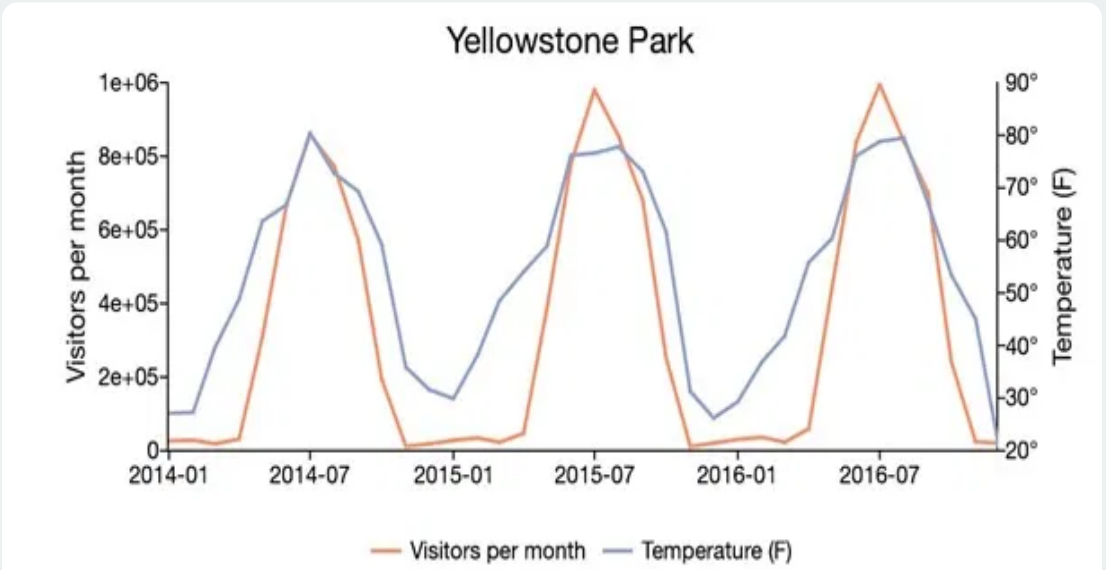
```
df.resample('D').mean()
```

↑ Upsampling

Aumenta a frequência dos dados (ex: de diário para horário). Requer **interpolação** para preencher os novos valores.

```
df.resample('H').interpolate()
```

Código	Descrição	Exemplo
'D'	Diário	Precipitação diária
'W'	Semanal	Crescimento semanal da cultura
'M'	Mensal	Temperatura média mensal
'H'	Horário	Umidade do solo por hora



A re-amostragem é essencial para alinhar dados de diferentes fontes e frequências em análises agrícolas!

Re-amostragem: Downsampling

↓ Downsampling: Diminuindo a Frequência

O **downsampling** reduz a frequência dos dados (ex: de horário para diário), exigindo uma função de agregação para combinar os valores.

Métodos de Agregação:

`sum()` `mean()` `min()` `max()` `first()` `last()`

Aplicações na Engenharia Agrícola:

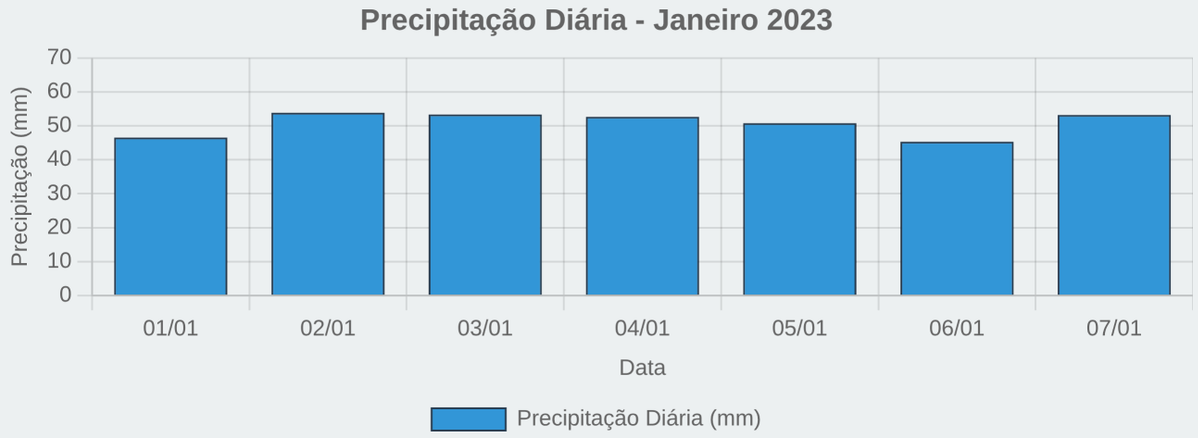
- ✓ Converter dados horários de precipitação para totais diários
- ✓ Calcular médias semanais de temperatura

Exemplo Prático

```
import pandas as pd
import numpy as np

# Criar DataFrame com dados de precipitação horária
idx = pd.date_range(start="2023-01-01 00:00",
                    periods=7*24, freq="h")  # 7 dias
precipitacao = pd.Series(np.random.rand(7*24) * 5,
                         index=idx)  # Precipitação em mm
```

```
Precipitação Diária (soma):
2023-01-01    55.129823
2023-01-02    63.881634
2023-01-03    63.649311
```



Re-amostragem: Upsampling

↑ Upsampling: Aumentando a Frequência

O **upsampling** aumenta a frequência dos dados (ex: de diário para horário), exigindo métodos de interpolação para preencher os novos valores.

Métodos de Interpolação:

linear | time | nearest | ffill

Aplicações na Engenharia Agrícola:

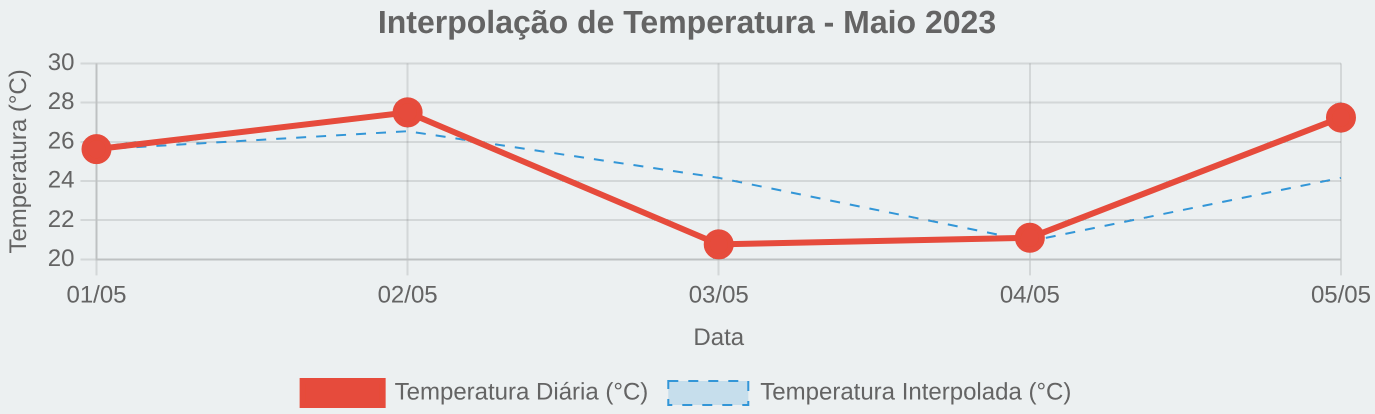
- ✓ Estimar valores horários de temperatura a partir de medições diárias

Exemplo Prático

```
import pandas as pd
import numpy as np

# Criar DataFrame com dados de temperatura diária
idx = pd.date_range(start="2023-05-01", periods=5, freq="D")
temperatura_diaria = pd.Series(np.random.uniform(20, 30, size=5),
                               index=idx)
```

```
Temperatura Horária (interpolada):
2023-05-01 00:00:00    26.589332
2023-05-01 01:00:00    26.680330
2023-05-01 02:00:00    26.771329
```



Aplicações em Engenharia Agrícola

Previsão de Safras

Análise de séries temporais de dados climáticos e histórico de produtividade para prever rendimentos futuros. Utiliza downsampling para agregar dados diários em médias semanais ou mensais.

Monitoramento de Culturas

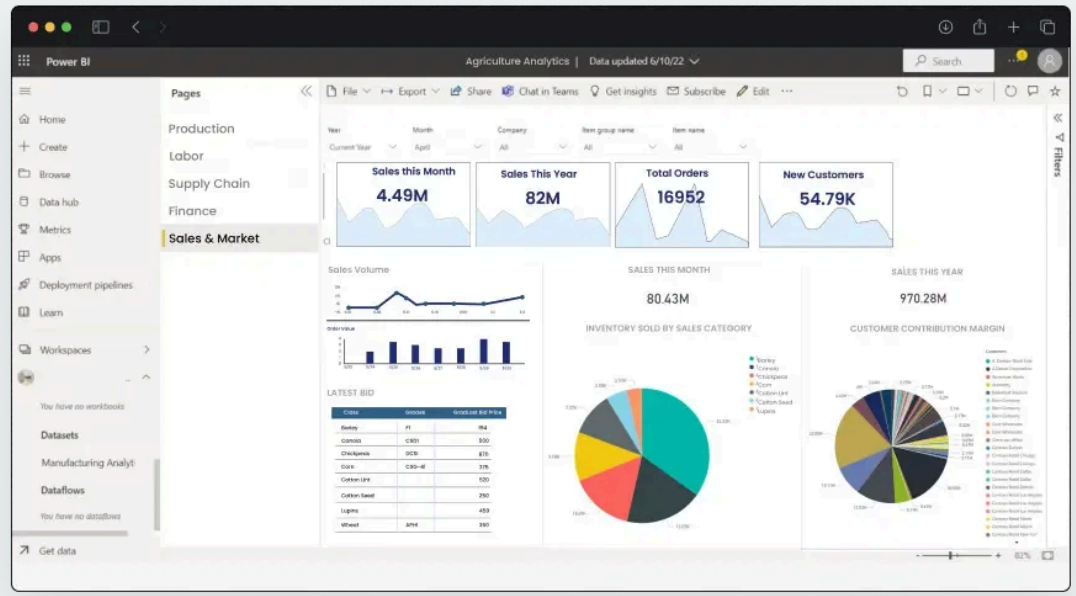
Processamento de séries temporais de imagens de satélite (NDVI) para acompanhar o desenvolvimento das culturas e detectar anomalias. Requer indexação temporal eficiente para comparar com anos anteriores.

Gestão de Irrigação

Análise de dados de umidade do solo e evapotranspiração para otimizar cronogramas de irrigação. Utiliza upsampling para estimar valores horários a partir de medições menos frequentes.

Previsão de Preços Agrícolas

Modelagem de séries temporais de preços de commodities para identificar tendências e sazonalidades, auxiliando na tomada de decisões de mercado.



Dashboard de análise de dados agrícolas com visualizações de séries temporais para monitoramento de culturas e condições climáticas.

Conclusão

Recapitulação

- **Tipos de Dados:** Timestamp, DatetimeIndex e Timedelta para trabalhar com séries temporais.
- **Conversão:** pd.to_datetime() converte diversos formatos para datas.
- **Criação:** pd.date_range() gera sequências de datas com frequência específica.
- **Indexação:** Seleção e filtragem eficiente de dados por períodos.
- **Downsampling:** Reduz a frequência dos dados com agregação.
- **Upsampling:** Aumenta a frequência dos dados com interpolação.

Obrigado!

Aplicações na Engenharia Agrícola

- **Previsão de Safras:** Análise de dados climáticos e histórico de produtividade.
- **Monitoramento de Culturas:** Processamento de imagens de satélite (NDVI).
- **Gestão de Irrigação:** Análise de dados de umidade do solo.

Próximos Passos

- → Explorar modelos de previsão (ARIMA, Prophet)
- → Analisar componentes de séries temporais
- → Integrar dados de múltiplas fontes
- → Desenvolver dashboards interativos