



Engenharia de Software
Dynamic Programming
Aula 5 – 2º semestre

Prof. Dr. Francisco Elânio

Conteúdo da aula

- Juntando dados de dataframes
- Uso do merge
- Análise exploratória dos dados
- Exemplos para responder perguntas sobre o canal
- Uso de estruturas de programação dinâmica

Unindo dados de dois dataframes

Exemplo do primeiro dataframe
Contendo ano, bloco, ndvi, area e
cdambiente

Exemplo do segundo dataframe
contendo ano, bloco, talhão,
corte e tch bloco

```
import pandas as pd


# DataFrame dados1
dados1 = pd.DataFrame({
    'Unnamed: 0': [0, 1, 2, 3],
    'ano': [2021, 2021, 2021, 2021],
    'bloco': ['604P0311', '111P1186', '116P1611', '116P1626'],
    'ndvi': [0.185666, 0.338386, 0.112711, 0.215624],
    'area': [20.63, 8.40, 7.94, 21.62],
    'cdambiente': ['A', 'B', 'B', 'D']
})

# DataFrame dados2
dados2 = pd.DataFrame({
    'ano': [2021, 2021, 2021, 2021],
    'bloco': ['604P0311', '111P1186', '116P1611', '116P1626'],
    'talhao': [5, 10, 6, 2],
    'corte': [4, 5, 18, 18],
    'tch_bloco': [70.75, 26.47, 95.64, 110.97]
})
```

Utilizando o Merge

Exemplo do primeiro dataframe
Contendo ano, bloco, ndvi, area e cdambiente

```
# Agrupar os DataFrames com base nas colunas 'ano' e 'bloco'  
df_merged = pd.merge(dados1, dados2, on=['ano', 'bloco'])  
df_merged
```

✓ 0.0s  Open 'df_merged' in Data Wrangler

	Unnamed: 0	ano	bloco	ndvi	area	cdambiente	talhao	corte	tch_bloco
0	0	2021	604P0311	0.185666	20.63	A	5	4	70.75
1	1	2021	111P1186	0.338386	8.40	B	10	5	26.47
2	2	2021	116P1611	0.112711	7.94	B	6	18	95.64
3	3	2021	116P1626	0.215624	21.62	D	2	18	110.97

+ Code


+ Markdown

Análise Exploratória dos dados

Quais as variáveis envolvidas?

```
import pandas as pd
```

```
df = pd.read_excel(r'C:\Users\Researcher182\Desktop\FIAP\1_2024\1 semestre\dynamic programming\dados_cana_açúcar.xlsx')  
df
```

✓ 2.5s  Open 'df' in Data Wrangler

	ano_safra	bloco	ndvi	area_tl	cdambiente	talhao	corte	tch_bloco
0	2021	604P0311	0.185666	20.63	A	5	4	70.75
1	2021	111P1186	0.338386	8.40	B	10	5	26.47
2	2021	116P1611	0.112711	7.94	B	6	18	95.64
3	2021	116P1626	0.215624	21.62	D	2	18	110.97
4	2021	134P1520	0.222302	24.94	C	2	3	59.20
...
20015	2022	605P0301	0.156169	24.30	D	10	5	56.29
20016	2022	220P2046	0.136303	30.73	D	1	4	63.26
20017	2022	226P2678	0.114775	14.20	C	1	5	38.27
20018	2022	225P2864	0.136284	38.42	D	1	3	48.27
20019	2022	225P2564	0.091362	9.99	C	10	6	24.59

Análise Exploratória dos dados

```
df.info()
[3] ✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20020 entries, 0 to 20019
Data columns (total 8 columns):
#   Column      Non-Null Count  Dtype
---  -
0   ano_safr    20020 non-null  int64
1   bloco       20020 non-null  object
2   ndvi        20020 non-null  float64
3   area_tl     20020 non-null  float64
4   cdambiente  20020 non-null  object
5   talhao      20020 non-null  int64
6   corte       20020 non-null  int64
7   tch_bloco   20020 non-null  float64
dtypes: float64(3), int64(3), object(2)
memory usage: 1.2+ MB
```

Quais as variáveis envolvidas?

- ano, bloco, ndvi, area, cd ambiente, talhão, corte e TCH.

Existem informações faltantes? Existem dados faltantes?

- Não.

Qual o tipo de variável envolvida no problema?

- Inteiro, objeto, float.

Análise Exploratória dos dados

```
df_2021 = df[df['ano_safra'] == 2021]
```

```
df_2022 = df[df['ano_safra'] == 2022]
```

```
df_2021
```

✓ 0.0s Open 'df_2021' in Data Wrangler

	ano_safra	bloco	ndvi	area_tl	cdambiente	talhao	corte	tch_bloco
0	2021	604P0311	0.185666	20.63	A	5	4	70.75
1	2021	111P1186	0.338386	8.40	B	10	5	26.47
2	2021	116P1611	0.112711	7.94	B	6	18	95.64
3	2021	116P1626	0.215624	21.62	D	2	18	110.97
4	2021	134P1520	0.222302	24.94	C	2	3	59.20
...
13187	2021	602F0069	0.132217	24.56	D	36	2	121.84
13188	2021	221P3410	0.160627	39.34	E	4	3	31.10
13189	2021	222P2210	0.091482	17.04	D	5	8	67.00
13190	2021	222P2275	0.115201	10.88	C	5	5	49.64
13191	2021	221F0056	0.101916	8.47	D	3	2	76.40

13162 rows × 8 columns

Filtro para selecionar os dados para cada ano

- Neste caso, foi criado um filtro, gerando dois dataframes, df_2021 e df_2022.

Análise Exploratória dos dados

```
# Área total da planta para 2021
```

```
df_2021 = df[df['ano_safra'] == 2021]
```

```
area_21 = df_2021.groupby('ano_safra')['area_tl'].sum()
```

```
# Área total da planta para 2022
```

```
df_2022 = df[df['ano_safra'] == 2022]
```

```
area_22 = df_2022.groupby('ano_safra')['area_tl'].sum()
```

```
print(area_21)
```

```
print(area_22)
```

```
✓ 0.0s [icon] Open 'area_22' in Data Wrangler
```

```
ano_safra
```

```
2021    163386.37
```

```
Name: area_tl, dtype: float64
```

```
ano_safra
```

```
2022    88937.33
```

```
Name: area_tl, dtype: float64
```

Qual a área total da planta?

- 163386,37


- 88937,33

Análise Exploratória dos dados

Nível de NDVI por ambiente

```
df_2021_ndvi_media = df_2021.groupby('cdambiente')['ndvi'].mean()
df_2021_ndvi_mediana = df_2021.groupby('cdambiente')['ndvi'].median()
df_2021_ndvi_min = df_2021.groupby('cdambiente')['ndvi'].min()
df_2021_ndvi_max = df_2021.groupby('cdambiente')['ndvi'].max()

print(df_2021_ndvi_media)
print(df_2021_ndvi_mediana)
print(df_2021_ndvi_min)
print(df_2021_ndvi_max)
```

✓ 0.0s  Open 'df_2021_ndvi_max' in Data Wrangler

Isso mostrará os valores mínimos, máximos, médios e medianos por ambiente. O que dará uma idéia do que está acontecendo com a produção.

Análise Exploratória dos dados

Produção por tipo de corte. Agrupar os dados por corte e retonar a média da coluna ndvi.

```
df_corte_media = df_2021.groupby('corte')['ndvi'].mean()  
df_corte_mediana = df_2021.groupby('corte')['ndvi'].median()  
df_corte_media
```

✓ 0.0s Open 'df_corte_media' in Data Wrangler

corte	
0	0.121849
2	0.167913
3	0.170080
4	0.172878
5	0.181167
6	0.181808
7	0.184246
8	0.207476
9	0.192264
12	0.190410
18	0.187169

Name: ndvi, dtype: float64

Análise Exploratória dos dados

Qual o maior e menor TCH por corte?

```
df_tch_corte_max = df_2021.groupby('cdambiente')['tch_bloco'].max()  
df_tch_corte_min = df_2021.groupby('cdambiente')['tch_bloco'].min()
```

```
print(df_tch_corte_min)  
print(df_tch_corte_max)
```

✓ 0.0s Open 'df_tch_corte_max' in Data Wrangler

```
cdambiente  
A    18.57  
B     0.92  
C     0.33  
D     1.15  
E     9.04  
Name: tch_bloco, dtype: float64  
cdambiente  
A    630.06  
B   2361.39  
C   6200.60  
D    955.54  
E    129.82  
Name: tch_bloco, dtype: float64
```


Uso de programação dinâmica

Máxima Produção Acumulada por Talhão

```
# Inicializar um dicionário para armazenar o máximo acumulado por talhão
max_acumulado = {}

# Iterar sobre as linhas do DataFrame
for index, row in df_2021.iterrows():
    talhao = row['talhao']
    tch_bloco = row['tch_bloco']

    # Se o talhão já tiver sido registrado, acumular o máximo
    if talhao in max_acumulado:
        max_acumulado[talhao] = max(max_acumulado[talhao], max_acumulado[talhao] + tch_bloco)
    else:
        # Caso contrário, inicializar o acumulado
        max_acumulado[talhao] = tch_bloco

# Converter o dicionário em um DataFrame para análise posterior
df_max_acumulado = pd.DataFrame(list(max_acumulado.items()), columns=['Talhao', 'Max_TCH_Acumulado'])
df_max_acumulado
```

✓ 0.7s 🔗 Open 'df_max_acumulado' in Data Wrangler

	Talhao	Max_TCH_Acumulado
0	5	62660.31
1	10	22590.74

Uso de programação dinâmica

■ Comparação de TCH por Corte Usando Programação Dinâmica

```
# Escolha um talhão específico para análise
talhao_especifico = 2

# Inicializar um dicionário para armazenar as diferenças acumuladas por corte
diferenca_acumulada = {}

# Iterar sobre as linhas do DataFrame filtrando pelo talhão específico
for index, row in df_2021[df_2021['talhao'] == talhao_especifico].iterrows():
    corte = row['corte']
    tch_bloco = row['tch_bloco']

    # Se o corte já tiver sido registrado, acumular a diferença
    if corte in diferenca_acumulada:
        diferenca_acumulada[corte] = tch_bloco - diferenca_acumulada[corte]
    else:
        # Caso contrário, inicializar o acumulado
        diferenca_acumulada[corte] = tch_bloco

# Converter o dicionário em um DataFrame para análise posterior
df_diferenca_acumulada = pd.DataFrame(list(diferenca_acumulada.items()), columns=['Corte', 'Diferenca_Acumulada'])
df_diferenca_acumulada
```

✓ 0.1s 🔗 Open 'df_diferenca_acumulada' in Data Wrangler

	Corte	Diferenca_Acumulada
0	18	-221.18
1	3	1251.29
2	2	528.53

***“O que sabemos é uma gota; o
que ignoramos é um oceano.”
(Issac Newton)***

Referências

- John Paul Mueller / Luca Massaron, Algoritmos para leigos, Editora Alta books, 2018 - 1ª edição.
- Additya Y. Bhargava, Entendendo Algoritmos - um guia ilustrado para programadores e outros curiosos, Editora Novatec, 2018, 1ª edição.
- José Augusto N. G. Manzano / Jayr Figueiredo de Oliveira, Algoritmos: Lógica Para Desenvolvimento de Programação de Computadores, 29ª edição, 2019.
- Thomas H. Cormen / Charles E. Leiseson / Ronald L. Rivest / Clifford Stein, Algoritmos - Teoria e Prática, Editora Elsevier, 2012, 3ª edição.