Ada Tech [DS-PY-004] Técnicas de Programação I (PY)

Apresentação

Rudiney Casali:

- Doutorado em Física (UFSC)
- Especialista em Dados (Mackenzie)
- Lead Senior Data Scientist (Kyndryl)
- linkedin.com/in/rcasali/

Requerimentos

- Todos aqui estão familiarizados com a ferramenta Jupyter Notebook?

- Você pode editar seu JN no browser, e usando anaconda. Usaremos o editor de texto Visual Studio Code (NNF).

- Os encontros com a língua inglesa serão inevitáveis daqui pra frente.

Conteúdos

- Aula 1 (Expositiva e Prática): Revisão GIT;
- Aulas 2 e 3 (Expositiva e Prática): Numpy;
- Aulas 4 e 5 (Expositiva e Prática): Pandas;
- Aulas 6, 7 e 8 (Prática): Trabalho em EDA;
- Aula 9 (Prática): Apresentação do Trabalho em EDA e Rubrica.

Aulas 3 e 4 | Pandas

O Problema dos dados tabulados

Imagine que você precisa trabalhar com dados tabulados, realizando análises práticas e manipulação de dados do mundo real, que demandam uma visualização mais intuitiva que aquela oferecida pelos arranjos, com performance superior para dados com mais de 500k linhas e com maior empregabilidade industrial. Que ferramenta você poderia empregar para trabalhar com esses dados em python?

Pandas

Um pacote de manipulação de dados em Python para dados tabulares, dados na forma de linhas e colunas, também conhecidos como DataFrames. Você pode pensar em um DataFrame como uma planilha Excel.

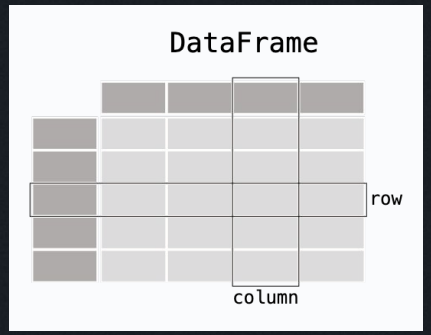
Pandas

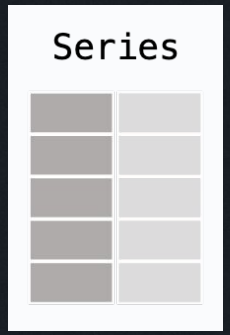
Funcionalidades incluem transformações de dados, como classificação de linhas e obtenção de subconjuntos, para calculo estatístico e remodelação/união de DataFrames. Funciona bem com outros pacotes populares de ciência de dados em Python, chamados de ecossistema PyData:

- Numpy
- Matplotlib, Seaborn, Plotly
- scikit-learn

Que tipo de dados os pandas tratam?

Para dados tabulares, como dados armazenados em planilhas ou bancos de dados, o pandas é a ferramenta certa. Ele permite explorar, limpar e processar seus dados. No pandas, uma tabela de dados é chamada de DataFrame.





Que tipo de dados os pandas tratam?

Primeiro importamos a **biblioteca** Pandas:

```
import pandas as pd
```

E criamos o **dataframe** df:

```
df = pd.DataFrame(
    {
        'Nome' : [
            "Silva, João",
            "Fernandes, Maria",
            "Silveira, Antônio",
        ],
        'Idade': [45, 34, 52],
        'Sexo': ['M', 'F', 'M'],
    }
)

df
```

Que tipo de dados os pandas tratam?

Primeiro importamos a **biblioteca** Pandas:

```
import pandas as pd
```

E criamos o **dataframe** df:

```
df = pd.DataFrame(
    {
        'Nome' : [
            "Silva, João",
            "Fernandes, Maria",
            "Silveira, Antˆnio",
        ],
        'Idade': [45, 34, 52],
        'Sexo': ['M', 'F', 'M'],
    }
)

df
```

Podemos definir uma **série** como uma coluna de um df:

```
df['Idade']
```

Ou podemos criar um **objeto** do tipo **série**:

```
Idade = pd.Series([45, 34, 52], name = 'Idade')
Idade
```

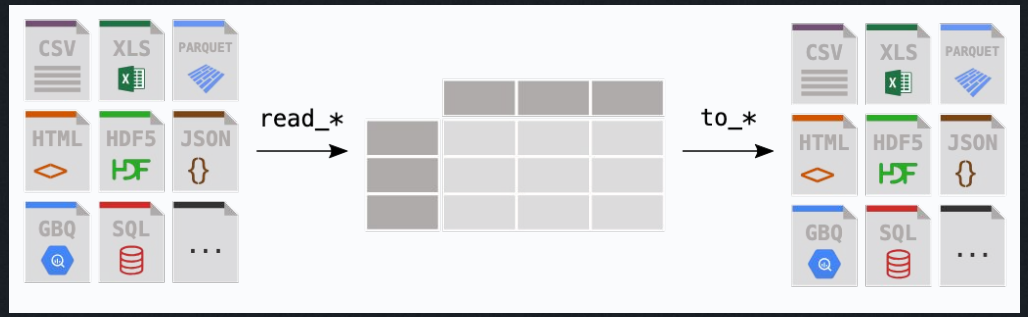
Podemos aplicar **atributos**

```
print(df['Idade'].max())
print(df['Idade'].min())
df.describe()
```

pandas

Como ler e escrever dados tabulares?

Suporta a integração com muitos formatos de arquivo ou fontes de dados prontos para uso (csv, excel, sql, json, parquet,…). A importação de dados de cada uma dessas fontes de dados é fornecida pela função com o prefixo read_*. Da mesma forma, os métodos to_* são usados para armazenar dados.



Como ler e escrever dados tabulares?

Algumas formas de se criar um dataframe:

```
dict = {
    'Id' : [1, 2, 3, 4, 5],
    'Nome' : [
        'Carlos Antônio', 'Maria da Silva',
        'Pedro Silveira', 'Paulo Pontes',
        'Gisele Fernandes',
        ],
    'Idade' : [49, 50, 45, 48, 57,],
    'Sexo' : ['M', 'F', 'M', 'M', 'F',],
    'Altura (m)' : [1.80, 1.65, 1.75, 1.73, 1.70,],
    'Peso (kg)' : [80, 65, 74, 73, 68,],
    }

df_from_pd = pd.DataFrame(data = dict)
df_from_pd
```

pandas

Como ler e escrever dados tabulares?

Algumas formas de se criar um dataframe:

```
dict = {
    'Id' : [1, 2, 3, 4, 5],
    'Nome' : [
        'Carlos Antônio', 'Maria da Silva',
        'Pedro Silveira', 'Paulo Pontes',
        'Gisele Fernandes',
        ],
    'Idade' : [49, 50, 45, 48, 57,],
    'Sexo' : ['M', 'F', 'M', 'M', 'F',],
    'Altura (m)' : [1.80, 1.65, 1.75, 1.73, 1.70,],
    'Peso (kg)' : [80, 65, 74, 73, 68,],
    }

df_from_pd = pd.DataFrame(data = dict)
df_from_pd
```

É possível ler de um arquivo csv:

```
df_from_csv = pd.read_csv('SYB66_1_202310_Population,
Surface Area and Density.csv', encoding = 'latin')
Df_from_csv
```

É possível também ler a partir de uma url:

```
url =
"https://data.un.org/_Docs/SYB/CSV/SYB66_1_202310_Pop
ulation,%20Surface%20Area%20and%20Density.csv"

df_from_url = pd.read_csv(url, encoding = 'latin')
df_from_url
```

pandas

Como ler e escrever dados tabulares?

Também é possível enviar um dataframe para um arquivo csv:

```
df_from_pd.to_csv(
    path_or_buf = 'df_from_pd.csv',
    index = False,
    columns = list(dict.keys()),
    )
```

Relendo o arquivo:

```
df_from_pd = pd.read_csv(
    filepath_or_buffer = 'df_from_pd.csv',
    encoding = 'utf-8',
    )
Df_from_pd
```

pandas

Como ler e escrever dados tabulares?

Também é possível enviar um dataframe para um arquivo csv:

```
df_from_pd.to_csv(
    path_or_buf = 'df_from_pd.csv',
    index = False,
    columns = list(dict.keys()),
    )
```

Relendo o arquivo:

```
df_from_pd = pd.read_csv(
    filepath_or_buffer = 'df_from_pd.csv',
    encoding = 'utf-8',
    )
Df_from_pd
```

Uma única coluna do dataframe define uma série:

```
idades = df_from_pd['Idade']
idades
```

O que pode ser comprovado com o comando type~:

```
type(df_from_pd['Idade'])
```

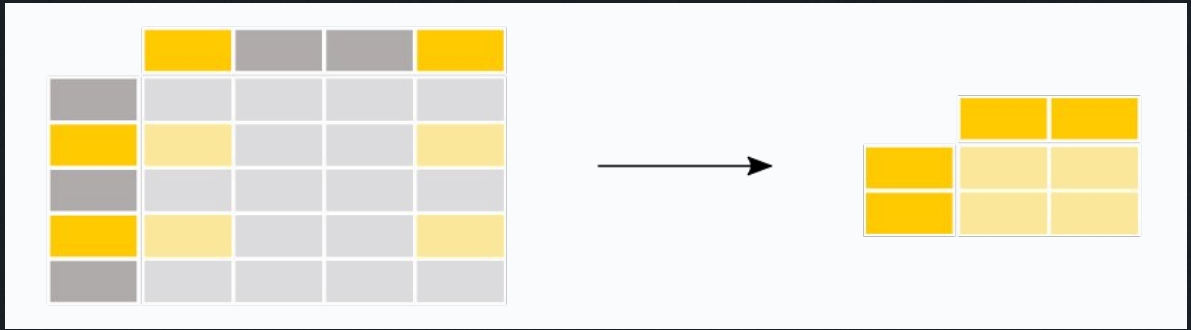
Cujo forma é:

```
df_from_pd['Idade'].shape
```

pandas

Como selecionar um subconjunto de uma tabela?

Métodos para fatiar, selecionar e extrair os dados necessários estão disponíveis no pandas.



Como selecionar um subconjunto de uma tabela?

Criamos novo dataframe a partir de um subconjunto das colunas:

```
Nome_Idade = df_from_pd[['Nome', 'Idade']]
Nome_Idade
```

E podemos checar o tipo:

```
type(Nome_Idade)
```

E a forma do objeto Nome_Idade:

```
Nome_Idade.shape
```

Podemos ainda aplicar filtros:

```
df_from_pd_50 = df_from_pd[df_from_pd['Idade'] > 50]
Df_from_pd_50
```

pandas

Como selecionar um subconjunto de uma tabela?

Criamos novo dataframe a partir de um subconjunto das colunas:

```
Nome_Idade = df_from_pd[['Nome', 'Idade']]
Nome_Idade
```

E podemos checar o tipo:

```
type(Nome_Idade)
```

E a forma do objeto Nome_Idade:

```
Nome_Idade.shape
```

Podemos ainda aplicar filtros:

```
df_from_pd_50 = df_from_pd[df_from_pd['Idade'] > 50]
Df_from_pd_50
```

E checamos a forma do objeto df_from_pd_50:

```
Df_from_pd_50.shape
```

Podemos filtrar o dataframe por sexo:

```
df_from_pd_M =
df_from_pd[df_from_pd['Sexo'].isin(['M'])]
print('df_from_pd_M.shape:', df_from_pd_M.shape)
df_from_pd_M
```

Podemos também filtrar por intervalos de altura:

```
df_from_pd_altura = df_from_pd[
    (df_from_pd['Altura (m)'] < 1.80) &
    (df_from_pd['Altura (m)'] > 1.70)]
print('df_from_pd_altura.shape:',
df_from_pd_altura.shape)
df_from_pd_altura
```

pandas

Como selecionar um subconjunto de uma tabela?

Podemos também filtrar por intervalos de altura:

```
df_from_pd_altura = df_from_pd[
   (df_from_pd['Altura (m)'] < 1.80) &
   (df_from_pd['Altura (m)'] > 1.70)]
print('df_from_pd_altura.shape:',
df_from_pd_altura.shape)
Df_from_pd_altura
```

Podemos também separar as mulheres em nossa base pelos nomes, com o uso do .loc:

```
names_F = df_from_pd.loc[
   df_from_pd['Sexo'] == 'F', 'Nome']
print('names_F.shape:', names_F.shape)
names_F
```

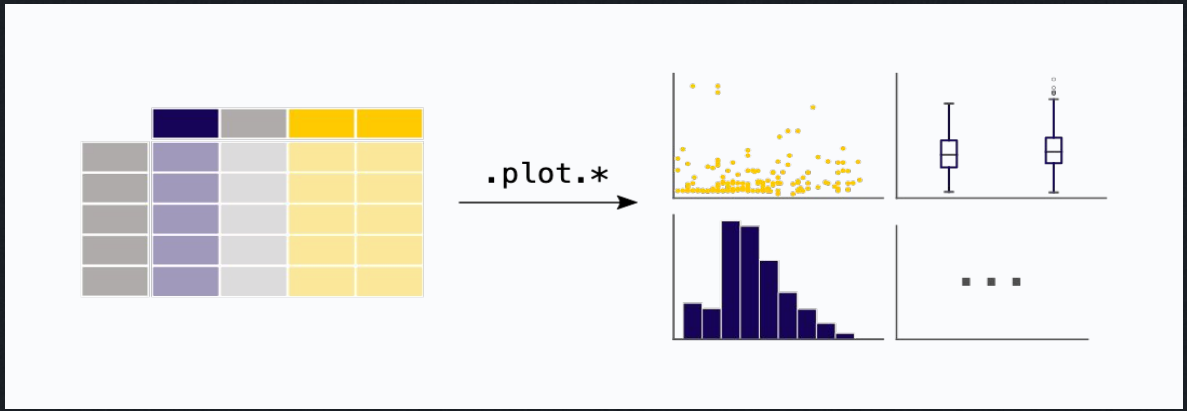
Podemos ainda corrigir um valor com o uso do .iloc

```
df_from_pd.iloc[0 : 1, 4] = 1.85
df_from_pd
```

pandas

Como traçar um gráfico em pandas?

O pandas fornece plotagem de seus dados prontos para uso, usando o poder do Matplotlib. Você pode escolher o tipo de gráfico (dispersão, barra, boxplot,…) correspondente aos seus dados.



Como traçar um gráfico em pandas?

Para traçar um gráfico você pode importar a biblioteca matplotlib:

```
import matplotlib.pyplot as plt
```

Usaremos o dados sobre dióxido de Nitrogênio (NO_2) em estações em Paris, Antuérpia e Londres.

```
air_quality = pd.read_csv(
    filepath_or_buffer = "air_quality_no2.csv",
    index_col = 0,
    parse_dates = True
    )
```

Usamos a atributo .plot()

```
air_quality.plot()
plt.show()
```

Como traçar um gráfico em pandas?

Para traçar um gráfico você pode importar a biblioteca matplotlib:

```
import matplotlib.pyplot as plt
```

Usaremos o dados sobre dióxido de Nitrogênio (NO_2) em estações em Paris, Antuérpia e Londres.

```
air_quality = pd.read_csv(
    filepath_or_buffer = "air_quality_no2.csv",
    index_col = 0,
    parse_dates = True
    )
```

Usamos a atributo .plot()

```
air_quality.plot()
plt.show()
```

Podemos focar em uma única coluna:

```
air_quality["station_paris"].plot()
plt.show()
```

Ou focar em uma dispersão com o atributo .scatter():

```
air_quality.plot.scatter(
    x = "station_london",
    y = "station_paris",
    alpha = 0.5
    )
plt.show()
```

Podemos ainda estudar os quartis das distribuições com o atributo .box():

```
air_quality.plot.box()
```

pandas

Como traçar um gráfico em pandas?

Também é possível traçar diferentes gráficos:

```
axs = air_quality.plot.area(
    figsize = (12, 4),
    subplots = True
    )

plt.show()
```

pandas

Como traçar um gráfico em pandas?

Também é possível traçar diferentes gráficos:

```
axs = air_quality.plot.area(
    figsize = (12, 4),
    subplots = True
    )

plt.show()
```

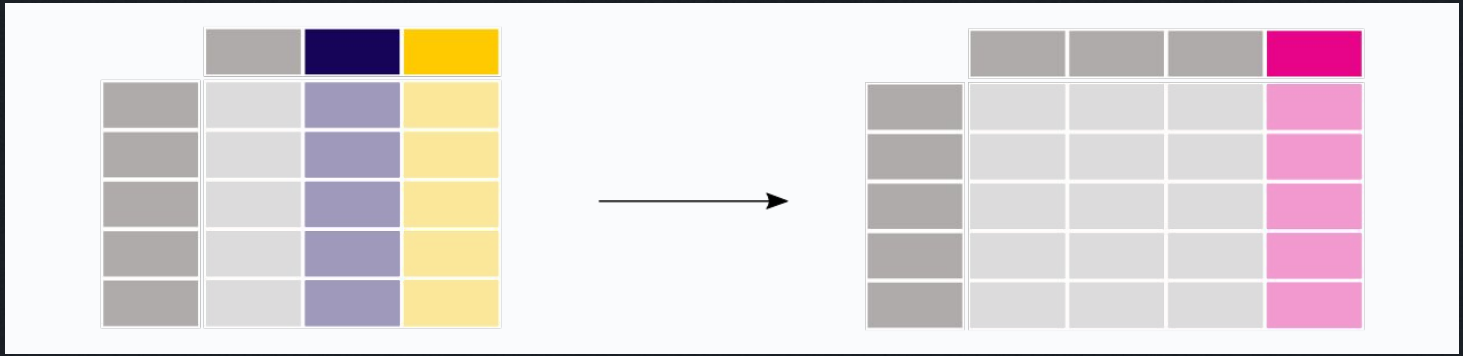
E criar uma arquivo png do gráfico traçado:

```
fig, axs = plt.subplots(figsize = (12, 4))

air_quality.plot.area(ax = axs)

axs.set_ylabel("Concentração de NO$_2$")

fig.savefig("Concentração de NO2.png")

plt.show()
```

pandas

Como criar novas colunas derivadas de colunas existentes?

As manipulações de dados em uma coluna funcionam elemento a elemento. Adicionar uma coluna a um DataFrame com base nos dados existentes em outras colunas é simples.



Como criar novas colunas derivadas de colunas existentes?

Podemos criar novas colunas a partir de operações com colunas originais:

```
titanic = pd.read_csv(
    filepath_or_buffer = "titanic.csv",
    index_col = 0,
    parse_dates = True
    )

sinking_year = 1912

titanic['year_born'] = sinking_year - titanic['Age']
titanic
```

pandas

Como criar novas colunas derivadas de colunas existentes?

Podemos criar novas colunas a partir de operações com colunas originais:

```
titanic = pd.read_csv(
    filepath_or_buffer = "titanic.csv",
    index_col = 0,
    parse_dates = True
    )

sinking_year = 1912

titanic['year_born'] = sinking_year - titanic['Age']
titanic
```

Com um dicionário de mapeamento:

```
dict = {
    'Survived' : 'sobrevivente',
    'Pclass' : 'Classe',
    'Name' : 'Nome',
    'Sex' : 'Sexo',
    'Age' : 'Idade',
    'SibSp' : 'Número_Irmãos_Cônjuge',
    'Parch' : 'Número_Pais_Filhos',
    'Ticket' : 'Bilhete',
    'Fare' : 'Tarifa',
    'Cabin' : 'Cabine',
    'Embarked' : 'Local_Embarque',
    'year_born' : 'Ano_Nascimento',
    }
```

Como criar novas colunas derivadas de colunas existentes?

Podemos renomear as colunas:

```
titanic_renamed = titanic.rename(
    columns = dict,
        )
```

E renomeamos o índice também:

```
titanic_renamed.rename_axis(
    mapper = 'IdPassengeiro',
    inplace = True
    )
titanic_renamed
```

pandas

Como criar novas colunas derivadas de colunas existentes?

Podemos renomear as colunas:

```
titanic_renamed = titanic.rename(
    columns = dict,
        )
```

E renomeamos o índice também:

```
titanic_renamed.rename_axis(
    mapper = 'IdPassengeiro',
    inplace = True
    )
titanic_renamed
```

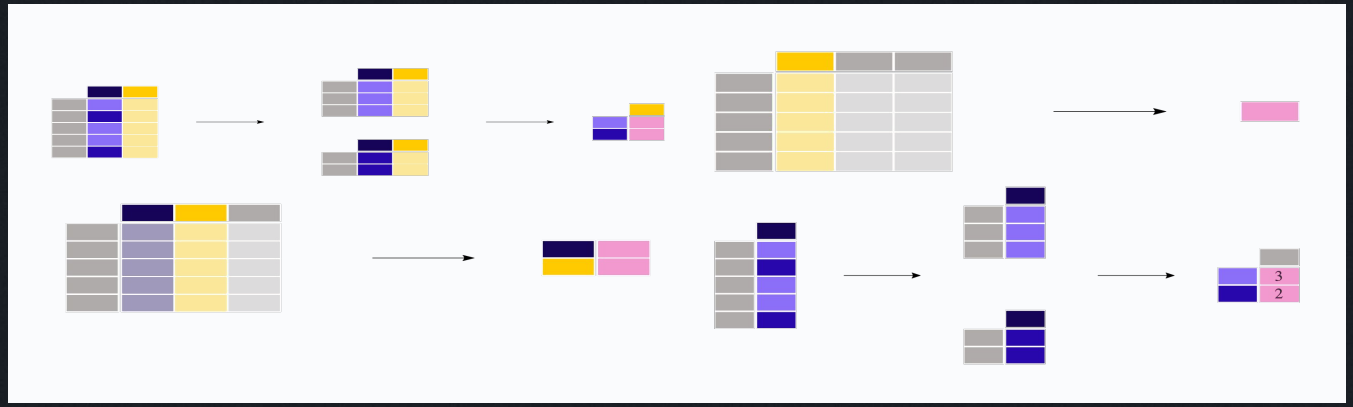
Podemos ainda editar os nomes:

```
titanic_renamed = titanic_renamed.rename(
    columns = str.lower
    )
titanic_renamed
```

pandas

Como calcular estatísticas resumidas?

Agregações padrões (média, mediana, mínimo, máximo, contagens…) ou personalizadas podem ser aplicadas em todo o conjunto de dados, em uma janela deslizante dos dados ou agrupadas por categorias. Esta última também é conhecida como abordagem split-apply-combine.



Como calcular estatísticas resumidas?

Podemos calcular a idade média e mediana dos passageiros:

```
print('idade média dos passageiros:',
      int(titanic_renamed[Idade].mean()))
print('idade mediana dos passageiros:',
      int(titanic_renamed[Idade].median()))
```

Ou podemos descrever o df com o atributo .describe()

```
titanic_renamed[[Idade, Tarifa, ]].describe()
```

Podemos ainda usar o atributo de agregação .agg()

```
titanic_renamed.agg(
    {
        "Idade": ["min", "max", "median", "skew"],
        "Tarifa": ["min", "max", "median", "mean"],})
```

pandas

ada

Como calcular estatísticas resumidas?

Podemos calcular a idade média e mediana dos passageiros:

```
print('idade média dos passageiros:',
     int(titanic_renamed[Idade].mean()))
print('idade mediana dos passageiros:',
     int(titanic_renamed[Idade].median()))
```

Ou podemos descrever o df com o atributo .describe()

```
titanic_renamed[[Idade, Tarifa, ]].describe()
```

Podemos ainda usar o atributo de agregação .agg()

```
titanic_renamed.agg(
   {
       "Idade": ["min", "max", "median", "skew"],
       "Tarifa": ["min", "max", "median", "mean"],})
```

Podemos ainda agrupar as idades por sexo, usando o atributo .groupby():

```
titanic_renamed[['Sexo', 'Idade']].groupby(by =
'Sexo').mean()
```

Ou também tomar a tarifa média, por sexo e classe:

```
titanic_renamed.groupby(
       ["Sexo", "Classe"])["Tarifa"].mean()
```

Também é possível contar o número de passageiros por categoria e sexo, usando o atributo .value_counts():

```
titanic_renamed[['Classe', 'Sex']].value_counts()
```

pandas

ada

Como calcular estatísticas resumidas?

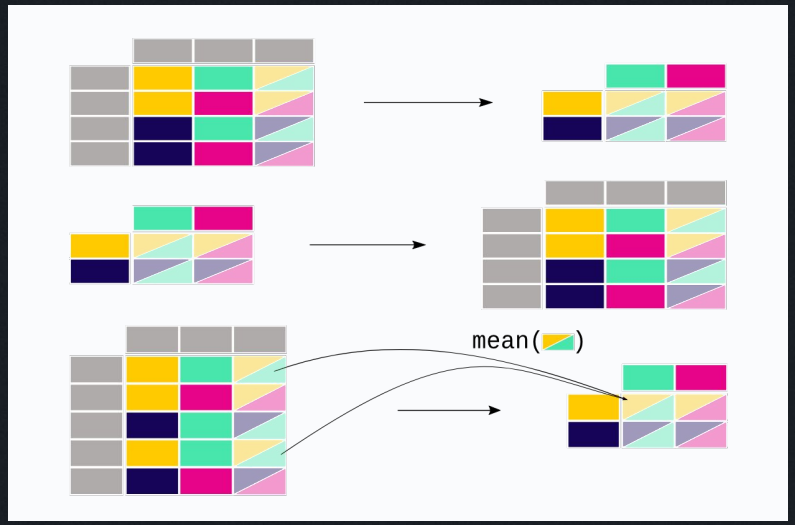
Ou simplesmente contar ocorrências, com o atributo `.count()`

```
titanic_renamed.groupby('Classe')['Classe'].count()
```

pandas

Como remodelar o layout das tabelas?

Altere a estrutura da sua tabela de dados de várias maneiras. Você pode derreter melt() sua tabela de dados do formato largo para o formato longo/organizado ou girar pivot() do formato longo para o formato largo. Com agregações integradas, uma tabela dinâmica é criada com um único comando.



pandas

Como remodelar o layout das tabelas?

Para ordenar um dataframe, usamos o atributo `.sort_values()`:

```
titanic_renamed.sort_values(
    by = ['idade', 'sexo'],
    ascending = False).head(50)
```

Podemos filtrar um df com o uso de uma máscara:

```
mask_male = titanic_renamed['sexo'] == 'male'
male = titanic_renamed[mask_male]
male
```

pandas

Como remodelar o layout das tabelas?

Para ordenar um dataframe, usamos o atributo .sort_values():

```
titanic_renamed.sort_values(
    by = ['idade', 'sexo'],
    ascending = False).head(50)
```

Podemos filtrar um df com o uso de uma máscara:

```
mask_male = titanic_renamed['sexo'] == 'male'
male = titanic_renamed[mask_male]
male
```

E selecionar os n = 20 primeiras ocorrências de cada local de embarque:

```
male_location =
male.sort_index().groupby(['local_embarque']).head(20
)
Male_location
```

Com o método .pivot() (long to wide) podemos observar as tarifas pagas, por cada passageiro, por sexo e local de embarque:

```
Titanic_renamed_pivot = titanic_renamed.pivot(
    columns = ['local_embarque', 'sexo'],
    values = 'tarifa')
```

pandas

Como remodelar o layout das tabelas?

O método .pivot_table() nos permite saber a tarifa média, por local de embarque, pago passageiro:

```
Titanic_renamed_pvt_tbl =
titanic_renamed.pivot_table(
    values = 'tarifa',
    index = 'local_embarque',
    columns = [
        'sobrevivente',
        'sexo',
        'classe',
        ],
    aggfunc = 'mean',
)
```

pandas

ada

Como remodelar o layout das tabelas?

O método .pivot_table() nos permite saber a tarifa média, por local de embarque, pago passageiro:

```
Titanic_renamed_pvt_tbl =
titanic_renamed.pivot_table(
    values = 'tarifa',
    index = 'local_embarque',
    columns = [
        'sobrevivente',
        'sexo',
        'classe',
        ],
    aggfunc = 'mean',
)
```

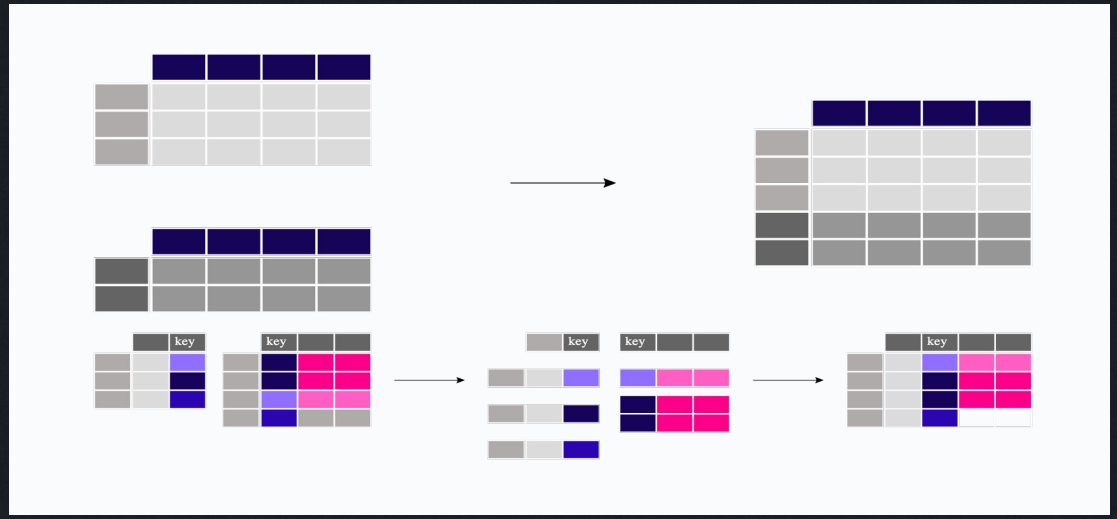
O método .melt() (wide to long) pode ser usado para extrairmos a idade de cada passageiro, por ordem do de sua identificação:

```
titanic_renamed.reset_index(inplace = True)
Titanic_renamed_melt = titanic_renamed.melt(
    id_vars = ['IdPassengeiro',],
    value_vars = ['idade'],
    value_name = 'd',
    var_name="id_location",
    )

Titanic_renamed_melt
```

pandas

Como combinar dados de multiplas tabelas?

Várias tabelas podem ser concatenadas tanto em colunas quanto em linhas, já que operações de junção join() / mesclagem merge() semelhantes a bancos de dados são fornecidas para combinar várias tabelas de dados.



Como combinar dados de multiplas tabelas?

Ao definirmos um novo dataframe:

```
dict = {
    'Id' : [1, 2, 3, 4, 5],
    'posição' : [
        'Operador de caixa', 'Gerente',
        'Supervisor', 'Encarregado', 'Contadora'],
    'Setor' : [
        'Atendimento', 'Atendimento',
        'Atendimento', 'Transportes',
'Contabilidade'],
    'Superior Direto' : [
        'Carlos Figueiredo', 'Anésia Freitas',
        'Paula Serrano', 'Ricardo Madeira', 'Carlos
Medeiros'],
    }

df_from_pd_2 = pd.DataFrame(data = dict)
df_from_pd_2
```

pandas

Como combinar dados de multiplas tabelas?

Ao definirmos um novo dataframe:

```
dict = {
   'Id' : [1, 2, 3, 4, 5],
   'posição' : [
       'Operador de caixa', 'Gerente',
       'Supervisor', 'Encarregado', 'Contadora'],
   'Setor' : [
       'Atendimento', 'Atendimento',
       'Atendimento', 'Transportes',
'Contabilidade'],
   'Superior Direto' : [
       'Carlos Figueiredo', 'Anésia Freitas',
       'Paula Serrano', 'Ricardo Madeira', 'Carlos
Medeiros'],
   }

df_from_pd_2 = pd.DataFrame(data = dict)
df_from_pd_2
```

Podemos agora usar o método `.concat()` para unir as tabelas df_from_pd e df_from_pd_2:

```
df_from_pd_3 = pd.concat(
   [df_from_pd.loc[:, 'Nome' : 'Idade'],
    df_from_pd_2.loc[:, 'posição' : 'Superior
Direto']
    ],
   axis = 1
   )
df_from_pd_3
```

pandas

ada

Como combinar dados de multiplas tabelas?

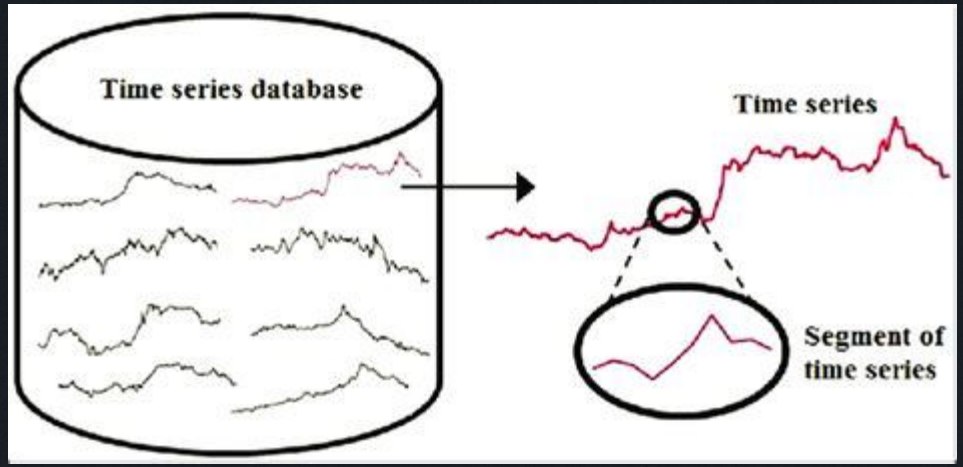
Com o método .merge() podemos fundir dois dataframes usando uma chave comum:

```
df_from_pd_4 = df_from_pd.merge(
    df_from_pd_2,
    left_on = 'Id',
    right_on = 'Id'
    )

df_from_pd_4
```

pandas

Como lidar com dados de série temporal?

Há suporte para séries temporais e um extenso conjunto de ferramentas para trabalhar com datas, horas e dados indexados por tempo.



pandas

Como lidar com dados de série temporal?

Vamos ler um novo arquivo e trabalhar com séries:

```
air_quality = pd.read_csv(
    filepath_or_buffer = 'air_quality_no2_long.csv',
    encoding = 'utf-8',)
```

Renomeamos uma coluna e a definimos como datetime:

```
air_quality = air_quality.rename(columns = {
'date.utc': 'datetime'})

air_quality['datetime'] =
pd.to_datetime(air_quality['datetime'])
```

Agora escolhemos uma cidade teste:

```
city_test = 'Paris'
```

pandas

Como lidar com dados de série temporal?

Vamos ler um novo arquivo e trabalhar com séries:

```
air_quality = pd.read_csv(
    filepath_or_buffer = 'air_quality_no2_long.csv',
    encoding = 'utf-8',)
```

Renomeamos uma coluna e a definimos como datetime:

```
air_quality = air_quality.rename(columns = {
'date.utc': 'datetime'})

air_quality['datetime'] =
pd.to_datetime(air_quality['datetime'])
```

Agora escolhemos uma cidade teste:

```
city_test = 'Paris'
```

E agrupamos por ela, tomando o valor médio das emissões de NO_2:

```
air_quality_Paris = air_quality[
    air_quality['city'] == city_test
    ].groupby(
    air_quality['datetime'].dt.hour
    )['value'].mean()

air_quality_Paris.plot(
    kind = 'bar',
    rot = 0,
    ax = axs
    )

plt.title(f'{city_test}')
plt.xlabel("Hour of the day");
plt.ylabel("$NO_2 (µg/m^3)$");
```

pandas

Como lidar com dados de série temporal?

Podemos definir uma das colunas como o índice:

```
no_2 = air_quality.pivot(
    index = 'datetime',
    columns = 'location',
    values = 'value'
    )
no_2.head()
```

Podemos tomar o intervalo total;

```
no_2[air_quality["datetime"].min() :
air_quality["datetime"].max()].plot();
```

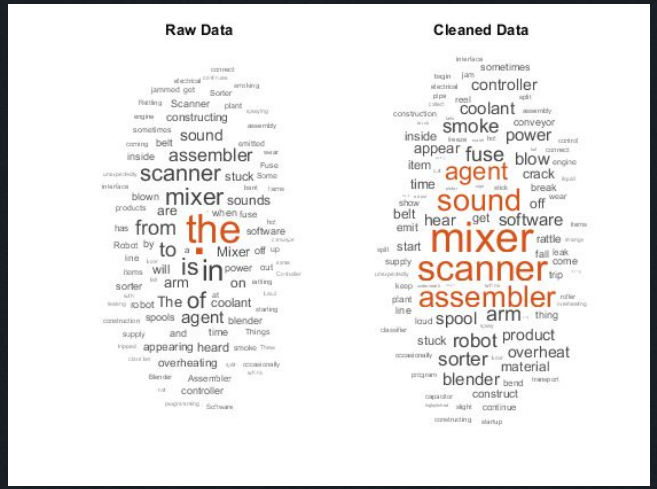
Ou apenas a variação em um dia:

```
no_2["2019-05-20" : "2019-05-21"].plot();
```

pandas

Como manipular dados textuais?

Pandas oferece uma ampla gama de funções para limpar dados textuais e extrair deles informações úteis.



Como manipular dados textuais?

Primeiro podemos formatar os nomes de passageiros começando por uma letra maiúscula, usando o atributo `.split()`:

```
titanic['Name'] = titanic['Name'].str.title()
```

Podemos também separar os nomes dos passageiros por vírgula (','):

```
titanic['Name'].str.split(",")
```

E criamos uma coluna para os sobrenomes, usando o atributo `.get()`:

```
titanic['Surname'] =
titanic['Name'].str.split(",").str.get(0)
titanic
```

pandas

Como manipular dados textuais?

Primeiro podemos formatar os nomes de passageiros começando por uma letra maiúscula, usando o atributo `.split()`:

```
titanic['Name'] = titanic['Name'].str.title()
```

Podemos também separar os nomes dos passageiros por vírgula (','):

```
titanic['Name'].str.split(",")
```

E criamos uma coluna para os sobrenomes, usando o atributo `.get()`:

```
titanic['Surname'] =
titanic['Name'].str.split(",").str.get(0)
titanic
```

Usando o atributo `.contains()`, podemos procurar por textos específicos nas columnas:

```
titanic['Name'].str.contains('Countess')
titanic['Name'].str.contains('Braund')
```

Ou a linha em que tal ocorrência se dá:

```
titanic[titanic['Name'].str.contains('Countess')]
```

Podemos contar quantos caracteres cada nome possui usando o atributo `.len()`:

```
titanic['Name'].str.len()
```

Podemos extrair o índice associado ao nome mais longo com o atributo `.idxmax()`:

```
titanic['Name'].str.len().idxmax()
```

pandas

ada

Como manipular dados textuais?

Podemos extrair o índice associado ao nome mais longo c
com o atributo .idxmax():

```
titanic['Name'].str.len().idxmax()
```

Por fim, podemos renomear as cidades de embarque com
o atributo .replace():

```
titanic['Embarked'] = titanic['Embarked'].replace({
    'C' : 'Cherbourg',
    'Q' : 'Queenstown',
    'S' :  'Southampton'
    })

titanic
```

pandas

Exercício proposto:

- Estude e realize os exercícios dos notebooks:
  - ADA_TECH_4_pandas_enunciado_exercicio_1.ipynb
  - ADA_TECH_5_pandas_enunciado_exercicio_2.ipynb
  - ADA_TECH_10_pandas_enunciado_exercicio_3.ipynb
  - ADA_TECH_11_pandas_enunciado_exercicio_4.ipynb

De volta ao problema dos dados tabulados

Pandas é uma ferramenta de análise e manipulação de dados de código aberto rápida, poderosa, flexível e fácil de usar, construído sobre a linguagem de programação Python.

Referências

- Pandas
- The pandas DataFrame: Make Working With Data Delightful
- Matplotlib: Visualization with Python
- Group by: split-apply-combine
- pandas.melt
- pandas.DataFrame.pivot
- Pandas DataFrame Concat Explained [6 Code Examples]
- pandas.DataFrame.join
- pandas.DataFrame.merge
- 3.12.1 Documentation » The Python Tutorial » 9. Classes
- Python Class Attributes
- Python Object Oriented Programming
- Understanding Libraries in Python: A Comprehensive Guide
- Tutorial: How to Create and Use a Pandas DataFrame
- Pandas – What is a Series Explained With Examples
- Objects in Python with Examples
- 14 Ways to Filter Pandas Dataframes

Referências

- Difference between loc() and iloc() in Pandas DataFrame
- Timeline and Facts About the Titanic
- A beginner's guide to Kaggle's Titanic problem
- Time series / date functionality
- Pivot vs Pivot Table Methods in Pandas. Definitions, Examples & Differences