Introdução ao Pandas

Patrícia Novais

Sumário

Pandas?

Pandas é uma biblioteca do Python muito utilizada em programação científica. Pandas trouxe um plus ao Python, pois possibilita trabalhar com análise de dados sem ter que recorrer a outras linguagens (como o R).

Pandas = PANel DAtaS

PANel DAta S

Pandas?

Mas o que o Pandas pode fazer por mim (por nós)?

O que podemos fazer com Pandas

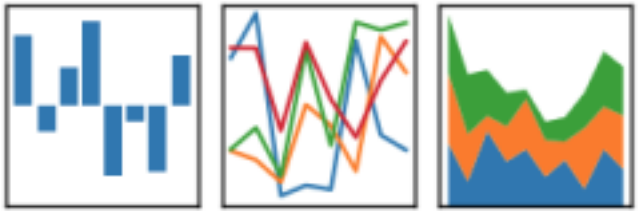
» **Manipulação de dados:** de forma rápida, ágil e com indexação integrada.

» **Análise de dados:** Leitura, escrita, alinhamento, reshaping, slicing, agrupamentos, fusão, concatenação…

» **Variedade de usos:** Mercado financeiro, Neurociência, Economia, Estatística, Publicidade(!) e muito mais...

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Vamos começar?

» Importando a biblioteca

```
[1]: import pandas as pd
```

⇒ Pandas não é uma biblioteca built-in, então é preciso instalá-la, caso a instalação do Python tenha sido feita sem o Anaconda.

Tipos de estruturas

» No Pandas podemos trabalhar com dois tipos de estruturas: *Series* e *Dataframe*.

» *Series*: um objeto unidimensional (*array*) que contém uma sequência de valores e seus respectivos índices. Podemos usar a função pd.Series() para criar uma *series*:

```
idades = pd.Series([15,21,30,12,24,52])
```

```
idades
```

```
0    15
1    21
2    30
3    12
4    24
5    52
dtype: int64
```

Como parâmetro, colocamos uma lista de valores

Tipos de estruturas
- *Series*

» Podemos também dizer qual o índice queremos para a Series:

```
idades2
```

```
Joao     15
Maria    21
Ana      30
Pedro    12
dtype: int64
```

» Para confirmar os índices:

```
idades2.index
```

```
Index(['Joao', 'Maria', 'Ana', 'Pedro'], dtype='object')
```

» Para obter apenas os valores:

```
idades2.values
```

```
array([15, 21, 30, 12], dtype=int64)
```

Tipos de estruturas
- *Dataframe*

» Outra importante estrutura, o *Dataframe* é uma tabela de dados com diversas colunas, que podem ser de diferentes tipos (numérico, categórico, booleano...)

» Podemos criar dataframes a partir da função pd.DataFrame(), utilizando como argumento tanto um array quanto usando dicionários.

```
» df = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]),columns=['a', 'b', 'c'])
```

```
df
```

	a	b	c
0	1	2	3
1	4	5	6
2	7	8	9

Tipos de estruturas - *Dataframe*

» Usando dicionários:

```
dados = {'valor1': [1, 2], 'valor2': [3, 4], 'valor3':[5,6]}
df2 = pd.DataFrame(data=dados)
```

```
df2
```

	valor1	valor2	valor3
0	1	3	5
1	2	4	6

» Usando dicionários e explicitando os índices:

```
dados = {'valor1': [1, 2], 'valor2': [3, 4], 'valor3':[5,6]}
df3 = pd.DataFrame(data=dados, index=['a','b'])
```

```
df3
```

	valor1	valor2	valor3
a	1	3	5
b	2	4	6

Tipos de estruturas
- *Dataframe*

» Similar ao caso da *Series*, podemos acessar tanto os valores quanto os índices do Dataframe com os métodos *pd.values* e *pd.index*:

```
df3.values
```

```
array([[1, 3, 5],
       [2, 4, 6]], dtype=int64)
```

```
df3.index
```

```
Index(['a', 'b'], dtype='object')
```

» Ainda podemos obter os nomes das colunas com o *pd.columns*:

```
df3.columns
```

```
Index(['valor1', 'valor2', 'valor3'], dtype='object')
```

Lendo um arquivo

» Outra maneira de se criar um dataframe é através da leitura de um arquivo.

» Pandas é capaz de ler diversos tipos de arquivos, com uma sintaxe muito simples. Dentre os tipos de arquivos que podemos ler com Pandas, temos:

- read_table
- *read_csv*
- *read_excel*
- *read_hdf*
- *read_sql*
- *read_json*
- *read_html*
- *read_stata*
- *read_sas*

Exemplos de leitura

» Pandas é capaz de ler diversos tipos de arquivos, com uma sintaxe muito simples. Veja alguns exemplos:

[2]:
```
df = pd.read_csv('bank-full.csv')

df = pd.read_table('bank-full.txt', delim_whitespace=True)

df = pd.read_excel('bank-full.xlsx', sheet_name='Dados')
```

Há diversos parâmetros que podem ser adicionados ao comandos de leitura... Veja a documentação de cada um sempre que precisar!

Manipulando o Dataframe
Head e Tail

» Podemos verificar o cabeçalho do dataframe com o comando *df.head()*:

```
[3]: df.head()
```

[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campai
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	

Manipulando o Dataframe
Head e Tail

» Podemos verificar o final do dataframe com o comando *df.tail()*:

```
[4]: df.tail()
```

[4]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361

Manipulando o Dataframe
Nomeando as colunas

» Podemos usar o *df.columns* para modificar todos os nomes das colunas:

```
[4]: df.columns = ['idade', 'job', 'estado civil', 'educacao', 'default', 'balance', 'housing',
                   'emprestimo', 'contato','dia', 'mes', 'duracao', 'campanha', 'pdays','previous',
                   'poutcome']
```

» Ou modificar o nome de colunas específicas usando a função *df.rename()*:

```
[6]: df.rename(columns={"idade": "Idade", "job": "Profissao"}, inplace=True)
```

Manipulando o Dataframe
Info

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 16 columns):
Idade           45211 non-null int64
Profissao       45211 non-null object
estado civil    45211 non-null object
educacao        45211 non-null object
default         45211 non-null object
balance         45211 non-null int64
housing         45211 non-null object
emprestimo      45211 non-null object
contato         45160 non-null object
dia             45211 non-null int64
mes             45211 non-null object
duracao         45200 non-null float64
campanha        45211 non-null int64
pdays           45211 non-null int64
previous        45211 non-null int64
poutcome        45211 non-null object
dtypes: float64(1), int64(6), object(9)
memory usage: 5.5+ MB
```

» É essencial entendermos o tipo de dado que temos em mãos. O comando *df.info()* nos ajuda a verificar os tipos das nossas variáveis e, inclusive, se há valores faltantes/nulos.

As variáveis *contato* e *duração* possuem dados faltantes/nulos

Manipulando o Dataframe
Describe: estatísticas básicas

» A função *df.describe()* exibe as estatísticas básicas dos dados numéricos.

```
[10]: df.describe()
```

[10]:

	Idade	balance	dia	duracao	campanha	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

Manipulando o Dataframe
Describe: estatísticas básicas

» Para estatísticas de variáveis categóricas, precisamos adicionar o argumento *include=['O']*.

```
[11]: df.describe(include=['O'])
```

[11]:

	Profissao	estado civil	educacao	default	housing	emprestimo	contato	mes	poutcome
count	45211	45211	45211	45211	45211	45211	45211	45211	45211
unique	12	3	4	2	2	2	3	12	4
top	blue-collar	married	secondary	no	yes	no	cellular	may	unknown
freq	9732	27214	23202	44396	25130	37967	29285	13766	36959

top: o valor que mais aparece
freq: frequência do valor mais comum (top)

Manipulando o Dataframe
Describe: estatísticas básicas

» Ainda, é possível aplicar o *pd.describe* a todos os dados:

```
df.describe(include='all')
```

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	
count	45211.000000	45211	45211	45211	45211	45211.000000	45211	45211	45160	45211.0
unique	NaN	12	3	4	2	NaN	2	2	3	
top	NaN	blue-collar	married	secondary	no	NaN	yes	no	cellular	
freq	NaN	9732	27214	23202	44396	NaN	25130	37967	29285	
mean	40.936210	NaN	NaN	NaN	NaN	1362.272058	NaN	NaN	NaN	15.8
std	10.618762	NaN	NaN	NaN	NaN	3044.765829	NaN	NaN	NaN	8.3
min	18.000000	NaN	NaN	NaN	NaN	-8019.000000	NaN	NaN	NaN	1.0
25%	33.000000	NaN	NaN	NaN	NaN	72.000000	NaN	NaN	NaN	8.0
50%	39.000000	NaN	NaN	NaN	NaN	448.000000	NaN	NaN	NaN	16.0
75%	48.000000	NaN	NaN	NaN	NaN	1428.000000	NaN	NaN	NaN	21.0
max	95.000000	NaN	NaN	NaN	NaN	102127.000000	NaN	NaN	NaN	31.0

Manipulando o Dataframe
Exercícios

1. Abra o arquivo **world_happiness_report_2015.csv** e o aloque em um dataframe.
2. Verifique o cabeçalho e o final do dataframe.
3. Quais as colunas desse dataframe?
4. Quais os tipos de dados temos no dataframe?
5. Há valores faltantes ou nulos? Em quais colunas?
6. Renomeie as variáveis como segue:

happiness rank	**=>**	rank_felicidade
happiness score	**=>**	score_felicidade
standard error	**=>**	stand_error
economy (GDP per Capita	**=>**	PIB
health (Life Expectancy)	**=>**	expect_vida
trust (Government Corruption)	**=>**	corrupcao

7. Quais os valores médios de **expect_vida**? E o valor mediano? E o máximo da variável **PIB**?
8. Crie uma *series* que contenha a altura de 5 colegas e deixe seus nomes como índice.

Manipulando o Dataframe
Selecionando dados

» Podemos acessar as informações dos dataframes e series de diversas maneiras.

» Quando queremos apenas 1 coluna:

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

```
0    58
1    44
2    33
3    47
```

ou

```
df.Idade
```

```
0    58
1    44
2    33
3    47
```

» Quando queremos mais de 1 coluna, podemos usar uma lista de variáveis

```
df[['Idade','dia','Profissao']]
```

	Idade	dia	Profissao
0	58	5	management
1	44	5	technician
2	33	5	entrepreneur

Manipulando o Dataframe
Seleção: iloc e loc

» Os métodos *iloc* e *loc* são utilizados para selecionar dados de um dataframe, mas possuem diferenças importantes.

» **iloc:** seleção baseadas nas posições dos índices das linhas e colunas (inteiros)

» **loc:** seleção baseadas nos nomes das variáveis

» Em ambos os casos, os argumentos do método são as linhas e as colunas de interesse.

```
df.loc[<linhas>,<colunas>]
```

```
df.iloc[<linhas>,<colunas>]
```

Seleção: iloc

» O iloc faz a seleção através dos valores inteiros dos índices, por um array ou ainda por fatias dos dados.

```
df.iloc[1]
```

```
Idade                44
Profissao       technician
estado civil        single
educacao         secondary
default                 no
balance                 29
housing                yes
emprestimo              no
contato                NaN
dia                      5
mes                    may
duracao                151
campanha                 1
pdays                   -1
previous                 0
poutcome           unknown
Name: 1, dtype: object
```

```
df.iloc[[1]]
```

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	duracao	campa
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151.0	

Note a diferença entre os resultados. Embora os valores sejam os mesmos, a apresentação é diferente

Seleção: iloc

» Seleção de linhas

```
df.iloc[[1]]
```

Apenas uma linha

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	duracao	campa
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151.0	

```
df.iloc[[-3]]
```

Apenas uma linha

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	duracao	c
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127.0	

```
df.iloc[1:3]
```

Um fatiamento de linhas

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	duracao	car
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151.0	
2	33	entrepreneur	married	secondary	no	2	yes	yes	NaN	5	may	76.0	

Seleção: iloc

» Seleção de colunas

```
df.iloc[:,1]
```

```
0           management
1           technician
2         entrepreneur
3          blue-collar
4              unknown
5           management
6           management
7         entrepreneur
8              retired
9           technician
10              admin.
11              admin.
12          technician
13          technician
```

Apenas uma coluna

```
df.iloc[:,1:3]
```

	Profissao	estado civil
0	management	married
1	technician	single
2	entrepreneur	married
3	blue-collar	married
4	unknown	single
5	management	married
6	management	single
7	entrepreneur	divorced

Um fatiamento de colunas

Seleção: iloc

» Seleção de linhas e colunas

```
df.iloc[1,0:3]
```

Uma linha e um fatiamento de colunas

```
Idade                   44
Profissao          technician
estado civil           single
Name: 1, dtype: object
```

```
df.iloc[0:2,0:3]
```

Uma fatiamento de linhas e colunas

	Idade	Profissao	estado civil
0	58	management	married
1	44	technician	single

```
df.iloc[[0,5,8],0:3]
```

Um array e um fatiamento de colunas

	Idade	Profissao	estado civil
0	58	management	married
5	35	management	married
8	58	retired	married

Seleção: loc

» Selecionando apenas os valores de uma linha:

```
df.loc[[1]]
```

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	duracao	campa
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151.0	

```
df.loc[1]
```

```
Idade              44
Profissao      technician
estado civil       single
educacao        secondary
default                no
balance                29
housing               yes
emprestimo             no
contato               NaN
dia                     5
mes                   may
duracao               151
campanha                1
pdays                  -1
previous                0
poutcome          unknown
Name: 1, dtype: object
```

Similar ao que observamos com o *iloc*

Seleção: loc

» Selecionando uma lista de linhas:

```
df.loc[[0,1,2]]
```

Um array de linhas

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	duracao	ca
0	58	management	married	tertiary	no	2143	yes	no	NaN	5	may	261.0	
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151.0	
2	33	entrepreneur	married	secondary	no	2	yes	yes	NaN	5	may	76.0	

» Selecionando uma fatia dos dados:

```
df.loc[0:2]
```

Uma fatia de linhas

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	duracao	ca
0	58	management	married	tertiary	no	2143	yes	no	NaN	5	may	261.0	
1	44	technician	single	secondary	no	29	yes	no	NaN	5	may	151.0	
2	33	entrepreneur	married	secondary	no	2	yes	yes	NaN	5	may	76.0	

Seleção: loc

» Selecionando uma lista de linhas e colunas:

```
df.loc[[0,1,],['Idade','Profissao','default']]
```

Uma lista de linhas e uma lista de colunas

	Idade	Profissao	default
0	58	management	no
1	44	technician	no

» Selecionando uma fatia de linhas e uma lista de colunas:

```
df.loc[0:2,['Idade','Profissao','default']]
```

Uma fatia de linhas e uma lista de colunas

	Idade	Profissao	default
0	58	management	no
1	44	technician	no
2	33	entrepreneur	no

Seleção: loc

» Selecionando com base em *uma condição*:

Uma condição

```
df.loc[df.Idade > 55]
```

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	durac
0	58	management	married	tertiary	no	2143	yes	no	NaN	5	may	26
8	58	retired	married	primary	no	121	yes	no	NaN	5	may	5
13	58	technician	married	unknown	no	71	yes	no	NaN	5	may	7

» Selecionando com base em *mais de uma condição*:

Duas ou mais condições

```
df.loc[(df.Idade > 55) & (df.Profissao == 'technician')]
```

	Idade	Profissao	estado civil	educacao	default	balance	housing	emprestimo	contato	dia	mes	duracao
13	58	technician	married	unknown	no	71	yes	no	NaN	5	may	71.0
30	57	technician	married	secondary	no	839	no	yes	unknown	5	may	225.0
35	57	technician	divorced	secondary	no	63	yes	no	unknown	5	may	242.0

Exercícios: iloc e loc

1. Selecione apenas os dados de **country**, **region**, **family** e **freedom** (usando loc)
2. Selecione apenas os dados de **country**, **region**, **family** e **freedom** (usando iloc)
3. Selecione apenas as primeiras 15 linhas de **country** e **PIB** (usando loc)
4. Selecione apenas as primeiras 15 linhas de **country** e **PIB** (usando iloc)
5. Selecione apenas os dados cujo **score_felicidade** seja maior que 5.
6. Selecione apenas os dados que sejam da *Southern Asia*.