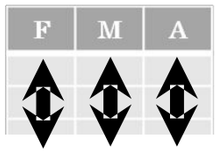
Data Wrangling

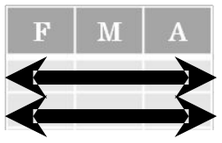
pandas Cheat Sheet
http://pandas.pydata.org

Organização de Dados Uma base para preparação de dados em pandas

Em um dataset organizado



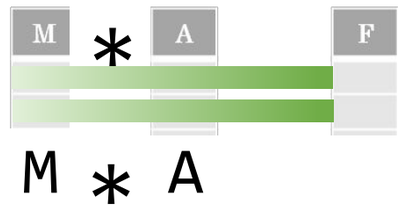
&



Cada **variável** é salva em sua própria **coluna**

Cada **observação** é salva em sua própria **linha**

Os dados organizados complementam as operações vetorizadas do pandas. O pandas preservará automaticamente as observações conforme você manipula as variáveis. Nenhum outro formato funciona tão intuitivamente com o pandas.



Criando DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4, 5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```
Especificando valores para cada coluna

```
df = pd.DataFrame(
     [[4, 7, 10],
      [5, 8, 11],
      [6, 9, 12]],
     index=[1, 2, 3],
     columns=['a', 'b', 'c'])
```
Especificando valores para cada linha

		a	b	c
N	v			
D	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
       [('d', 1), ('d', 2),
        ('e', 2)], names=['n', 'v']))
```
Criando DataFrame com um Multi Índice

Encadeamento

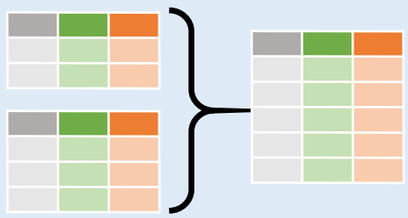
A maioria dos métodos em pandas retorna um data-frame para que outro método em pandas possa ser aplicado no resultado. Isso melhora a leitura do código

```
df = (pd.melt(df)
        .rename(columns={
                'variable':'var',
                'value':'val'})
        .query('val >= 200')
     )
```

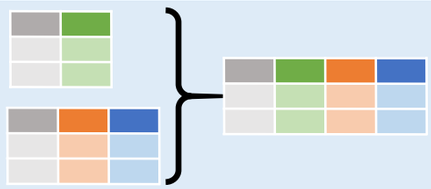
Remodelando Dados – Mudando layout, ordenação, reindexando, renomeando

pd.melt(df)
Junta as colunas em linhas

df.pivot(columns='var', values='val')
Espalha as linhas em colunas



pd.concat([df1,df2])
Concatena linhas de dataframes



pd.concat([df1,df2], axis=1)
Concatena colunas (axis=1) de dataframes.

df.sort_values('mpg')
Ordena as linhas pelo valor da coluna (crescente).

df.sort_values('mpg', ascending=False)
Ordena as linhas pelo valor da coluna (decrescente).

df.rename(columns = {'y':'year'})
Renomeia as colunas do dataframe

df.sort_index()
Ordena o index do dataframe

df.reset_index()
Reinicia o índice do DataFrame para linhas enumeradas, move o índice anterior para uma coluna

df.drop(columns=['Length', 'Height'])
Apaga colunas do DataFrame

Observando subconjuntos - linhas

df[df.Length > 7]
Extrai linhas que atendam aos critérios

df.drop_duplicates()
Remove linhas duplicadas

df.sample(frac=0.5)
Seleciona aleatoriamente uma amostra de linhas

df.sample(n=10) Seleciona n linhas

df.nlargest(n, 'value')
Seleciona e ordena as maiores entradas

df.nsmallest(n, 'value')
Seleciona e ordena as menores entradas

df.head(n)
Seleciona as primeiras n linhas

df.tail(n)
Seleciona as ultimas n linhas

Variáveis de subconjuntos-colunas

df[['width', 'length', 'species']]
Seleciona várias colunas específicas

df['width'] *or* **df.width**
Seleciona uma única coluna

df.filter(regex='*regex*')
Seleciona colunas que os nomes casem com a expressão regular (regex)

Usando query

query() permite expressões de verdadeiro ou falso (boolean) para filtrar linhas

df.query('Length > 7')

df.query('Length > 7 and Width < 8')

df.query('Name.str.startswith("abc")', engine="python")

Subconjuntos - linhas e colunas

Use **df.loc[]** e **df.iloc[]** para selecionar só linhas, só colunas ou ambos

Use **df.at[]** e **df.iat[]** para acessar um único valor de linha e coluna

Primeiro índice seleciona linhas, segundo colunas

df.iloc[10:20]
Seleciona linhas 10-20.

df.iloc[:, [1, 2, 5]]
Seleciona colunas nas posições 1, 2 e 5 (primeira coluna é 0).

df.loc[:, 'x2':'x4']
Seleciona todas as colunas entre x2 and x4 (inclusive).

df.loc[df['a'] > 10, ['a', 'c']]
Seleciona linhas que caem na condição lógica e somente colunas específicas

df.iat[1, 2] Acessa valores únicos pelo índice

df.at[4, 'A'] Acessa valores únicos pelo rótulo

Logica em Python (e pandas)

<	Menor que	!=	Diferente
>	Maior que	df.column.isin(*values*)	Está dentro da lista
==	Igual	pd.isnull(*obj*)	É nulo
<=	Menor ou igual	pd.notnull(*obj*)	Não é nulo
>=	Maior ou igual	&,|,~,^,df.any(),df.all()	Logico and, ou, not, xor, any, all

regex (Expressão regular) Exemplos

'\.'	Corresponde a strings contendo um ponto '.'
'Length$'	Corresponde a strings que terminam com a palavra'Length'
'^Sepal'	Corresponde a strings começando com a palavra'Sepal'
'^x[1-5]$'	Corresponde a strings começando com 'x'e terminando com 1,2,3,4,5
'^(?!Species$).*'	Corresponde a strings que não sejam 'Species'

Cheatsheet for pandas (http://pandas.pydata.org/ originally written by Irv Lustig, Princeton Consultants, inspired by Rstudio Data Wrangling Cheatsheet

Resumo dos Dados

df['w'].value_counts()
Contq o número de linhas com cada valor único da variável

len(df)
de linhas no DataFrame.

df.shape
Tuple da # de linhas, # de colunas no DataFrame.

df['w'].nunique()
de valores únicos da coluna.

df.describe()
Descrição básica e estatística sobra cada coluna (ou GroupBy).

pandas provê uma grande quantidade de **funções de sumarização** que opera em diferentes tipos de objetos do panda (DataFrame colunas, Series, GroupBy (veja abaixo)) e produz valores únicos para cada coluna ou grupo. Quando aplicado ao DataFrame, o resultado é retornado como uma Serie para cada coluna. Exemplos:

sum()
Soma os valores de cada objeto

count()
Conta não-NA/nulos valores de cada objeto.

median()
Mediana de cada objeto.

quantile([0.25,0.75])
Quantis de cada objeto.

apply(*function*)
Aplica uma função (function).

min()
Valor mínimo de cada objeto.

max()
Valor máximo de cada objeto.

mean()
Média de cada objeto.

var()
Variação de cada objeto.

std()
Desvio padrão de cada objeto.

Agrupar Dados

df.groupby(by="col")
Retorna o objeto GroupBy, agrupando pelos valores na coluna "col".

df.groupby(level="ind")
Retorna o objeto GroupBy, agrupando pelos valores no nível de indice "ind".

Todas as funções de sumarização acima podem ser aplicadas a grupos
Funções adicionais de GroupBy:

size()
Tamanho de cada grupo.

agg(*function*)
Agrega o grupo usando uma função

Os exemplos abaixo também podem ser aplicados a grupos. Nesse caso, a função é aplicada por grupo e os vetores retornados têm o comprimento do DataFrame original.

shift(1)
Copie com valores deslocados em 1.

rank(method='dense')
Classificações sem lacunas.

rank(method='min')
Rankeia.Os empates têm min. pto

rank(pct=True)
Rankeia dimensionadando entre [0, 1].

rank(method='first')
Rankeia. Empates vão para o primeira

shift(-1)
Copie com valores defasados em 1.

cumsum()
Soma cumulativa

cummax()
Maximo cumulativo

cummin()
Mínimo cumulativo

cumprod()
Produto cumulativo

Janelamento

df.expanding()
Retorna um objeto de expansão permitindo que funções de resumo sejam aplicadas cumulativamente

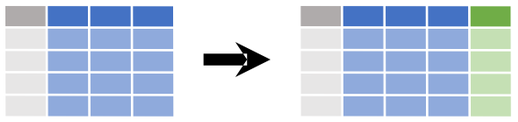
df.rolling(n)
Retorna um objeto de móvel/rolante permitindo que funções de resumo aplicado a janelas de comprimento n.

Lidando com Valores Faltantes

df.dropna()
Elimine linhas com qualquer coluna com dados NA/nulos.

df.fillna(value)
Substitua todos os dados NA/nulos por valor “value”.

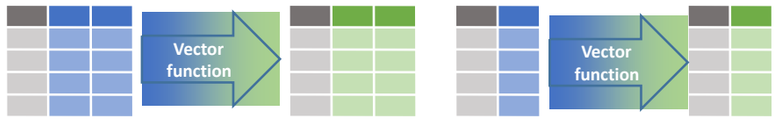
Gerando novas Colunas



df.assign(Area=lambda df: df.Length*df.Height)
Calcule e anexe uma ou mais novas colunas.

df['Volume'] = df.Length*df.Height*df.Depth
Adicione uma única coluna.

pd.qcut(df.col, n, labels=False)
Coluna de compartimento em n buckets.



O pandas fornece um grande conjunto de funções vetoriais que operam em t
colunas de um DataFrame ou em uma única coluna selecionada (uma série d
pandas). Essas funções produzem vetores de valores para cada uma das colu
uma única série para a série individual. Exemplos:

max(axis=1)
Max por elemento

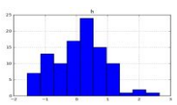
clip(lower=-10,upper=10)
Cortar valores nos limites de entrada

min(axis=1)
Min por elemento

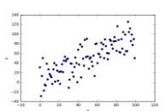
abs()
Valor absoluto

Gráficos

df.plot.hist()
Histograma de cada coluna



df.plot.scatter(x='w',y='h')
Scatter usando pares de pontos



Combinando Dados

adf

x1	x2
A	1
B	2
C	3

+

bdf

x1	x3
A	T
B	F
D	T

=

Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

pd.merge(adf, bdf, how='left', on='x1')
Junte as linhas correspondentes de bdf a adf.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

pd.merge(adf, bdf, how='right', on='x1')
Junte as linhas correspondentes de adf a bdf.

x1	x2	x3
A	1	T
B	2	F

pd.merge(adf, bdf, how='inner', on='x1')
Junta dados. Retenha apenas linhas em ambos os conjuntos.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

pd.merge(adf, bdf, how='outer', on='x1')
Junta dados. Retenha todos os valores, todas as linhas.

Filtering Joins

x1	x2
A	1
B	2

adf[adf.x1.isin(bdf.x1)]
Todas as linhas em adf que estão em bdf.

x1	x2
C	3

adf[~adf.x1.isin(bdf.x1)]
Todas as linhas em adf que não estão em bdf.

ydf

x1	x2
A	1
B	2
C	3

+

zdf

x1	x2
B	2
C	3
D	4

=

Set-like Operations

x1	x2
B	2
C	3

pd.merge(ydf, zdf)
Linhas que aparecem em ambos ydf e zdf (Intersecção).

x1	x2
A	1
B	2
C	3
D	4

pd.merge(ydf, zdf, how='outer')
Linhas que aparecem em um ou em ambos ydf e zdf (União).

x1	x2
A	1

pd.merge(ydf, zdf, how='outer', indicator=True)
.query('_merge == "left_only"')
.drop(columns=['_merge'])
Linhas que aparecem em ydf, mas não em zdf (Setdiff).

Cheatsheet for pandas (http://pandas.pydata.org/) originally written by Irv Lustig, Princeton Consultants, inspired by Rstudio Data Wrangling Cheatsheet