Estatística Aplicada à Computação/Telemática

Tutorial Básico ao Pandas

In [ ]:
```
import pandas as pd
import numpy as np
```

INTRODUÇÃO

As unidades básicas de trabalho com Pandas são as **'Series'** e os **'DataFrames'**.

Series nada mais são do que um conjunto de elementos de 1 dimensão com índices. Series podem conter basicamente qualquer objeto: int , float, string, etc... A forma mais fácil para criar uma series é através da função *pd.Series(data=, index=):*

In [ ]:
```
# usando uma lista
s1 = pd.Series (data = [5, 6, 8, -3, 9])
print (s1, '\n')

# usando um dicionário
d = {'Campeão': 'Flamengo', 'Vice':'Vasco', 'Terceiro':'Bangu'}
s2 = pd.Series(d)
print (s2, '\n')

# usando um numpy array
dados = np.random.random(4)
indices = np.arange(0, 8, 2)
s3 = pd.Series(data = dados, index = indices)
print(s3, '\n')

# usando uma variável escalável
s4 = pd.Series(-9, index=['a', 'b', 'c', 'd'])
print (s4)
```

Exercise 1 Qual a diferença entre uma lista, um dicionário, um array numpy e uma series pandas?

Exercise 2 Verifique a saída de cada uma das séries criadas acima, explicando as diferenças entre as formas de criação.

Exercise 3 Como modificar os índices (index) de uma série depois que ela foi criada?

Exercise 4 Como mudar os valores de uma série após ela ser criada?

DataFrames:

são simplesmente um conjunto de séries que compartilham o mesmo índice. Similiar a uma tabela ou planilha do excel. É objeto Pandas mais utilizado! Por isso é muito importante dominar todas as operações que envolvam dataframes. Existem várias formas de criar dataframes, vamos ver as mais importantes:

1) Através da função: *pd.DataFrame(data= , index= , columns=)*

In [ ]:
```
# Data aceita diferentes inputs:

# Dicionário
d = {'Times Paulistas': ['São Paulo', 'Santos', 'Corinthians'],
     'Times Cariocas': ['Flamengo', 'Fluminense', 'Botafogo'],
     'Times Ruins': ['Palmeiras', 'Vasco', 'Remo']}
df = pd.DataFrame(d)
print(df, '\n')

# Arrays
dados = np.random.random((5,3))
df2 = pd.DataFrame(data = dados, index = np.arange(5), columns = np.random.randint(1, 100, 3))
print (df2, '\n')

# Listas ou Tuplas
listas = [[9,3,6], [6,2,9], [6,1,6]]
df3 = pd.DataFrame(listas, index = (2,3,4), columns = ['A', 'B', 'C'])
print (df3)
```

Exercise 5 Como alterar índices, valores e nome de colunas de um dataframe? Dê um exemplo usando um dos criados acima.

Exercise 6 Como juntar as duas séries a seguir em um único dataframe?

In [ ]:
```
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))
```

2) Por leitura de arquivo de diferentes formatos com os métodos:
pd.read_csv(); pd.read_excel(); pd.read_pickle(); pd.read_sql(); pd.read_json(); pd.read_html() e outros...

In [ ]:
```
# basta passar a localização do arquivo para criar um novo dataframe

df = pd.read_csv('https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv')
df.head()      #  o método head() retorna as primeiras cinco linhas do dataframe
```

Veja que só passamos como argumento 'iris.csv', isso porque o arquivo csv estava na mesma pasta que o nosso código, caso contrário seria necessário passar todo o caminho, exemplo: "c:/users/exemplo/pasta/arquivo.csv". O dataset (conjunto de dados) iris.csv foi baixado de kaggle.com e contém características de três espécies de flores.

Todos os outros métodos são bem semelhantes, o que muda um pouco é o read_html(). Veja o exemplo:

In [ ]:
```
# Criamos um dataframe chamado fut e passamos o link que contém a tabela que queremos:
fut = pd.read_html("https://pt.wikipedia.org/wiki/Campeonato_Carioca_de_Futebol")
print("Quantidade de tabelas:", len(fut))
```

Perceba que o Pandas pega todas as tabelas presentes na página, como só queremos uma delas, selecionamos a escolhida:

In [ ]:
```
fut = fut[5]
fut.head()
```

Output:

Pandas dá a opção de salvar nosso df em diferentes formatos: (análogo aos métodos de leitura)

```
to_csv(); to_excel(), to_pickle(), to_json(), to_sql(), to_html e outros...
```

In [ ]:
```
# Aquela tabela que lemos no formato .html, podemos salvar assim:

fut.to_excel('Tabela dos Campeões.xlsx')           # Excel
fut.to_csv('Tabela dos Campeões.csv')              # Arquivo .csv
fut.to_pickle('Tabela dos Campeões.pickle')        # Pickle
```

Atributos e Funções:

Podemos obter muitas informações dos nossos dataframes, para isso basta conhecer as principais funções disponíveis:

.head() e .tail()

In [ ]:
```
# Criamos nosso dataframe
df = pd.read_csv('https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv')

# Como já vimos, para retornar as n primeiras usamos:
print(df.head(4))   # se não passar nenhum valor, o padrão é 5

# Para ver as últimas linhas do nosso df:
print(df.tail(3))
```

In [ ]:
```
# Para obter as colunas de seu dataframe:

print (df.columns)

# Para obter os índices de seu dataframe:

print (df.index)
```

Para obter informações rápidas, use os métodos .info() e .describe()

In [ ]:
```
# Retorna quantidade de linhas, colunas, tipo de dados...

df.info()
```

Para ser aplicado nas colunas (lembre que cada coluna de um df é um Series):

In [ ]:
```
# Para saber quais são os valores únicos que determinada coluna possui, use .unique()

df['species'].unique()
```

In [ ]:
```
# Existem duas formas de saber a quantidade de valores únicos:

print(len(df['species'].unique()))
print(df['species'].nunique())
```

In [ ]:
```
# Outro método importante é o .value_counts() , retorna a contagem de cada valor presente na coluna:

df['species'].value_counts()
```

Seleção e Indexação:

In [ ]:
```
# Dataframe de Exemplo
df = pd.DataFrame({'Times_Paulistas': ['São Paulo', 'Santos', 'Corinthians'],
     'Times_Cariocas': ['Flamengo', 'Fluminense', 'Botafogo'],
     'Times_Ruins': ['Palmeiras', 'Vasco', 'Remo']}, index = ['Campeão', 'Vice', 'Terceiro'])
df
```

Existem duas formas de selecionar colunas do seu df:

In [ ]:
```
df.Times_Paulistas

# ou, a minha preferência pessoal:

df['Times_Paulistas']
```

In [ ]:
```
# Para pegar mais de uma coluna, basta:

df[['Times_Paulistas', 'Times_Cariocas']]    # perceba que estamos passando uma lista de colunas ao invés de uma co
                                             # por isso os colchetes duplos... Retorna um dataframe!
```

Existem duas formas para selecionar linhas: pela posição ou pelo nome do index

In [ ]:
```
# Pela posição, com iloc[]:

df.iloc[0]
```

In [ ]:
```
# Pelo nome do índice:

df.loc['Vice']
```

In [ ]:
```
# Para escolher uma única célula, use os rótulos da coluna e da linha:

df['Times_Ruins']['Vice']
```

In [ ]:
```
# Adicionar colunas é muito simples

df['Nova Coluna'] = [1,2,3]
```

Índices:

In [ ]:
```
# Para mudar o índice - escolhemos como índice a nova coluna

df.set_index('Nova Coluna')
```

ATENÇÃO!!! A maioria das mudanças que fizermos nos dataframes não é permanente, na verdade o Pandas cria um novo objeto e retorna o mesmo. Isso evita a perda de informação acidental. Para que a mudança seja permanente, use o parâmetro *inplace* = *True*

In [ ]:
```
# Nesse caso não tem sentido mudar o índice, mas veja que a mudança não é permanente

df
```

In [ ]:
```
# Para resetar o índice, fazendo com que a mudança seja permanente

df.reset_index(inplace = True)
df
```

SELEÇÃO CONDICIONAL:

In [ ]:
```
data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'year': [2012, 2012, 2013, 2014, 2014],
        'reports': [4, 24, 31, 2, 3],
        'coverage': [25, 94, 57, 62, 70]}
df = pd.DataFrame(data, index = ['Cochice', 'Pima', 'Santa Cruz', 'Maricopa', 'Yuma'])
df
```

Vendo somente as linhas em que cobertura é maior que 50

In [ ]:
```
df[df.coverage > 50]
```

Vendo as linhas em que coverage é maior que 50 e reports menos que 4

In [ ]:
```
df[(df.coverage  > 50) & (df.reports < 4)]
```

Outros exemplos

In [ ]:
```
raw_data = {'first_name': ['Jason', 'Molly', np.nan, np.nan, np.nan],
        'nationality': ['USA', 'USA', 'France', 'UK', 'UK'],
        'age': [42, 52, 36, 24, 70]}
df = pd.DataFrame(raw_data, columns = ['first_name', 'nationality', 'age'])
df
```

In [ ]:
```
# Create variable with TRUE if nationality is USA
american = df['nationality'] == "USA"

# Create variable with TRUE if age is greater than 50
elderly = df['age'] > 50

# Select all cases where nationality is USA and age is greater than 50
df[american & elderly]
```

In [ ]:
```
# Select all cases where the first name is not missing and nationality is USA
df[df['first_name'].notnull() & (df['nationality'] == "USA")]
```

Aplicando funções e Operações nos dataframes:

In [ ]:
```
df = pd.DataFrame({'cor': ['preto','preto','preto','branco','branco','branco'],
                        'letra': ['A', 'E', 'I', 'A', 'E', 'I'],
                        'num': [1, 2, 3, 4, 5, 6]})
df
```

In [ ]:
```
# Vamos usar a coluna 'num' para aplicar algumas funções

df['num']
```

In [ ]:
```
# Primeiro passo é escrever uma função

def dobrar(x):      # função simples que retorna o dobro do número passado
    return x * 2
```

In [ ]:
```
# Agora é só aplicar na coluna desejada com .apply(), passando a nossa função como argumento

df['num'] = df['num'].apply(dobrar)    # para tornar a alteração permanente, atribuímos o resultado a nossa coluna
```

In [ ]:
```
df['num']
```

Acredito que deu pra trazer pelo menos o básico da biblioteca Pandas nesse tutorial. Eu sei que é muita coisa, mas tem muito mais coisa ainda para ser estudado! Se quiser se aprofundar mais, leia a documentação oficial, links abaixo.

FONTES:

1 - http://pandas.pydata.org/ (http://pandas.pydata.org/)
2 - http://pandas.pydata.org/pandas-docs/stable/ (http://pandas.pydata.org/pandas-docs/stable/)
3 - http://pandas.pydata.org/pandas-docs/stable/10min.html#min (http://pandas.pydata.org/pandas-docs/stable/10min.html#min)
4 - http://pandas.pydata.org/pandas-docs/stable/api.html#general-functions (http://pandas.pydata.org/pandas-docs/stable/api.html#general-functions)

In [ ]:
```

```