**Programação Phyton / Anaconda**

import pandas as pd # importar biblioteca pandas

import requests # importar biblioteca para obter dados de URL

resposta = requests.get(url) # Obter dados da URL

resposta.text # Verificar texto obtido

df = pd.DataFrame (resposta.json()[ ‘X’ ]) # Converter a resposta em formato JSON para um DataFrame

table = pd.read\_html(resposta.text) # Converter para tabela a resposta em formato HTML

table # Verificar texto obtido

df = table [ X ] # Definir DataFrame de tabela HTML

df = pd.read\_csv ( ‘arquivo.csv ’ ) # Carregar o arquivo CSV

df = pd.read\_csv (‘arquivo ’, sep = ';' ) # Informar separador do arquivo csv

df = pd.read\_csv (‘arquivo ’, encoding='latin1') # Informar o encoding do arquivo

df = pd.read\_csv (‘arquivo ’, decimal =',') # Informar o separados de casas decimais do arquivo

df = pd.read\_csv (‘arquivo ’, usecols=[0,2,3,,5]) # Definir colunas a serem carregadas

df = pd.read\_csv (‘arquivo ’, skiprows=X) # Definir as X primeiras linhas a serem deletadas

df = pd.read\_csv (‘arquivo ’, skipfooters=X) # Definir as X últimas linhas a serem deletadas

df.rename ( columns = { 'Xxxx' : 'Yyyy' }, inplace=True) # Renomear coluna específica

df.columns = [ ' nome0 ', ' nome1 ', ' nome2 ', ... ] # Trocar o nome das colunas

df.columns = df.iloc [ 0 ] # Usar o nome da primeira linha nas colunas

df.drop ( df.index [ 0 ], inplace = True) # Apagando a primeira linha

df [ ‘coluna' ] = pd.to\_datetime ( df [ ‘coluna' ] ) # converter os dados de uma coluna para data

df [ ‘coluna' ] = df [ ‘coluna' ].astype ( int ) # converter os dados para inteiros

df [ ‘coluna' ] = df [ ‘coluna' ].str.replace ( ‘x’ , ‘y’ ) # substituir valores

df.head () # Mostrar as primeiras linhas

df.head () .T # Mostrar as primeiras linhas na transversal

df.tail () # Mostrar as últimas linhas

df.sample () # Mostrar linhas aleatórias

df.shape # Formato do DataFrame (linhas,colunas)

df.count () # Contar dados válidos das colunas

df.info () # Tipos e dados válidos das colunas

df.dtypes # Tipos de dados das colunas

df.describe () # Análise dos dados numéricos do DataFrame

df['coluna'].unique () # Lista os valores de uma coluna específica

df [‘coluna '].count () # Contar valores de uma coluna específica

df [‘coluna '].value\_counts () # Contar valores por item de uma coluna específica

df [‘coluna\_data'].dt.year.value\_counts() # Contar valores por ano de uma coluna de data

df [‘coluna '] .str.contains (' xxxx ') ] .count () # Filtrar a contagem por parte de uma palavra da coluna

df [‘coluna '].value\_counts (ascending=True) # Contar valores repetidos em ordem invertida

df\_xxxx = df [ df [ 'coluna' ] == ‘yyyyy'] # Criar um subconjunto a partir dos dados originais

df\_xxxx = df [ df [ 'coluna' ] > Y] # Criar um subconjunto a partir dos dados originais

df\_xxxx = df [ df [ ‘coluna' ] != 'yyyyyyyy'] .copy() # Criar um subconjunto excluindo determinado item de uma coluna (diferente: !=)

df [ (df [ 'coluna' ] > X ) & (df [ 'coluna' ] < Y ) ] # Mostrar os valores de um intervalo definido

df [ df [ 'coluna' ] == df [ 'coluna' ].max () ] # Trazer a linha com valor mais alto de uma coluna

df [ df [ 'coluna' ] == df [ 'coluna' ].min () ] # Trazer a linha com valor mais baixo de uma coluna

df.groupby ( ‘coluna1' )[ 'coluna2' ] .max () # Trazer o maior valor para cada item da coluna1

df.groupby ( ‘coluna1' )[ 'coluna2' ] .min () # Trazer o menor valor para cada item da coluna1

df.groupby ( ‘coluna1' )[ 'coluna2' ] .sum () # Trazer a soma para cada item da coluna1

df.groupby ( ‘coluna1' )[ 'coluna2' ] .mean () # Trazer a média para cada item da coluna1

df.groupby ( ‘coluna1' )[ 'coluna2' ] .std () # Trazer o desvio padrão para cada item da coluna1

df.groupby ( ‘coluna1' )[ 'coluna2' ] .median () # Trazer median para cada item da coluna1

...] .sum () .sort\_values (ascending = False ) # Apresentar os valores pela ordem de valores

...] .sum () .round(X) # Definir valores de casas decimais

df.iloc [ 0, 3 ] # Retornar um campo pelo índice [ linha, coluna ]

df.loc [ 0, 'xxxx' ] # Retornar o valor com base no nome da coluna

df.loc [ df.groupby ( ‘coluna1' )[ 'coluna2' ] .idxmax () ] # Trazer as linhas com o maior valor para cada item

df.loc [ df.groupby ( ‘coluna1' )[ 'coluna2' ] .idxmin () ] # Trazer as linhas com o menor valor para cada item

df.nlargest ( X ,'coluna' ) # Trazer as linhas com os X maiores valores

df.nsmallest ( X ,'coluna' ) # Trazer as linhas com os X menores valores

**Matplotlib**

Import matploylib.pyplot as plt ist# importar biblioteca

%matplotlib inline # Informar o matplotlib sobre gráficos inline

df [‘coluna '].value\_counts () .plot.bar () # Plotar os valores em um gráfico vertical

df [‘coluna '].value\_counts () .plot.barh () # Plotar os valores em um gráfico horizontal

df [lista de colunas] .sum().plot.barh() # Plotar gráfico de barra para cada coluna de dados

df [‘coluna '].value\_counts () .plot.bar (title=' Título ') # Adicionar um título ao gráfico

df [‘coluna '].value\_counts () .plot.hist () # Plotar os valores em um histograma

df [‘coluna '].value\_counts () .plot.hist (bins=X) # Definir quantidade de barras do histograma

plt.xticks ( rotation = xx ) # Entrar com o ângulo de rotação da legenda

plt.show() # Mostrar ajuste

plt.title (Xxxx Xxxxxx', fontsize=Y) # Inserir título e definir tamanho da fonte

plt.xlabel (Xxxx Xxxxxx', fontsize=Y) ou plt.ylabel # Inserir título do eixo e definir tamanho da fonte

plt.axvline (X) # Inserir linha vertical no valor “X”

plt.axhline( 1.0, color='black', linestyle='--') # Inserir linha horizontal, na cor preta e pontilhada

**Seaborn**

Import seaborn as sns # importar biblioteca

sns.pointplot ( x = 'xxxx' , y = 'yyyy , data=df ) # Exibir o gráfico com a marcação de cada ponto

sns.distplot ( df[‘x’] , bins=X , color='red', kde=False ) # Exibir histograma com X barras, vermelho e sem linha de aproximação

sns.swarmplot (data=df , x=’coluna1', y='coluna2') # Gráfico de distribuição de pontos por categoria

**Folium**

!pip install folium # pip permite instalar novas bibliotecas e ! permite a execução de comandos diretamente no jupyter notebook

Import folium # importar biblioteca

mapa = folium.Map ( location=[ -X , -Y ] , zoom\_start = 5 # Criar mapa a partir de uma coordenada central

# Percorrer o DataFrame com base na condição “ >= X ” dos valores de uma coluna e desenhar um marcador para cada item usando for e interrows:

for indice, nome in df [ df [ ‘coluna' ] >= X ] .iterrows ():

folium.Marker(

location=[nome [ 'latitude' ] , nome[ 'longitude' ] ],

popup=nome ['colunaX'],

icon=folium.map.Icon(color='red')

) .add\_to (mapa)

mapa # mostrar mapa

**Endereços para pesquisa**

http:// github.com/OSBrasilia

http:// github.com/marcosvafg

<https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html>

**Cursos**

Python para Zumbis