Disciplina: Matemática Especial

Profs.: *Eduardo Negri, Alexandre Fernandes, Victor D´Ávila*

Tema 7 – Pandas, Matplotlib e Seaborn

A utilização conjunta das bibliotecas **Pandas**, **Matplotlib**, e **Seaborn** (http://seaborn.pydata.org/tutorial.html) potencializam a análise e visualizam dos do conjunto de dados contidos nos DataFrames.

Nessa secção iremos retornar ao projeto SIMCOSTA (http://www.simcosta.furg.br/home), visto no **Tema 5**, para compreender um pouco melhor e exercitar as ferramentas disponíveis nessas três bibliotecas.

Basicamente, vamos apreender como:

- combinar dataframes,
- realizar estatística básica de correlação cruzada,
- plotar os resultados.

7.1 Combinando Dataframes

Os objetos **Series** e **DataFrame** são ferramentas poderosas para explorar e analisar dados. Parte dessa capacidade vem de uma abordagem multifacetada para combinar conjuntos de dados isolados.

Com o **Pandas**, você pode mesclar (***merge***), juntar (***join***) e concatenar (***concatenate***) conjuntos de dados, permitindo unificar e entender melhor os dados à medida que são analisados.

Há três formas disponíveis para combinar dados no **Pandas**:

- **merge ()** para combinar dados em colunas ou índices comuns
- **.join ()** para combinar dados em uma coluna-chave ou índice
- **concat ()** para combinar DataFrames em linhas ou colunas

Uma visão completa para utilização dessas funcionalidades do Pandas pode ser obtida no *site*: https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html

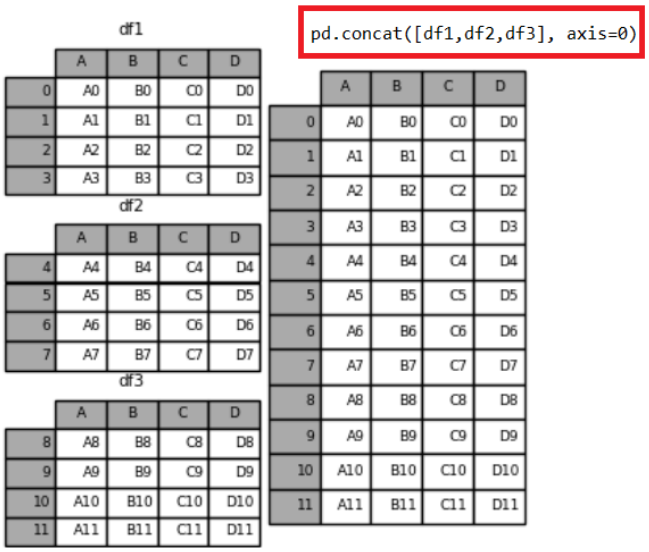
Aqui vamos tecer algumas considerações sobre a função **concat ()**, uma vez que iremos utilizá-la em nosso programa do projeto *SimCosta*.

A sintax da função `concat()` está descrita abaixo, observe que há várias *keywords* que poderão ser utilizadas, por exemplo: `axis` e `join`.

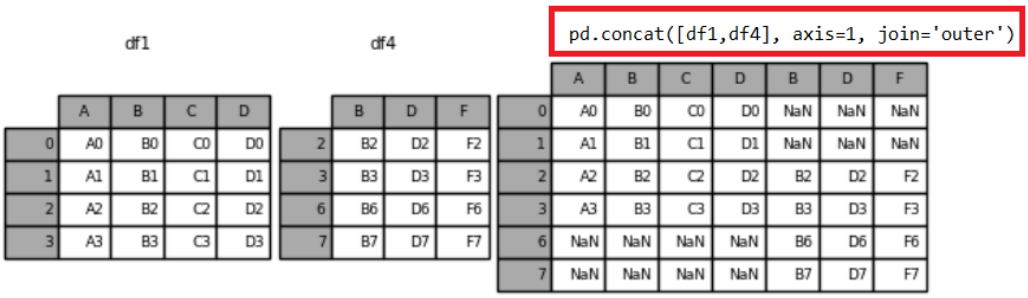
```
pd.concat(objs, axis=0, join='outer', ignore_index=False, keys=None,
          levels=None, names=None, verify_integrity=False, copy=True)
```

Os exemplos abaixo foram coletados do *site*: https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html

Uma concatenação simples dos DataFrames utilizando a *keyword* ***axis=0*** pode ser observada abaixo. Nesse caso, o alinhamento se dá pela sobreposição das linhas, uma vez que **axis=0** diz respeito às linhas. Note que as colunas dos DataFrames (df1, df2 e df3) tem o mesmo nome: A, B, C e D.

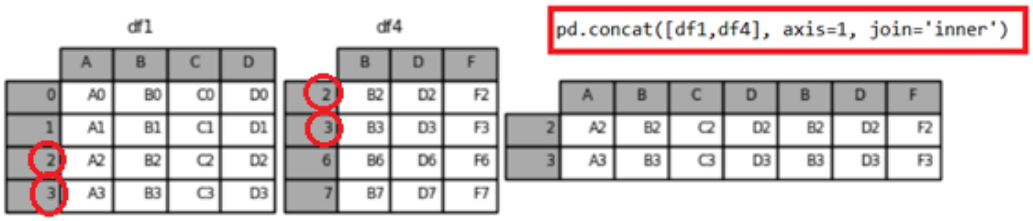


A seguir fazemos uso das *keywords* ***axis*** e ***join*** para realizar a concatenação. Observe que escolhendo o ***axis=1*** e ***join='outer'***, houve um alinhamento das colunas dos dataframes, uma vez que estamos utilizando o ***axis=1***. Onde não houve correspondência entre os índices dos DataFrames, os valores foram preenchidos com **NaN**, ou seja, com ***join='outer'*** nenhuma linha é descartada.



A concatenação realizada no exemplo abaixo, é igual a anterior, porém agora com a keyword ***join='inner'***. Note que houve um alinhamento das colunas do dataframes levando em consideração os índices, porém agora foram

descartadas as linhas onde os índices não tiveram correspondência entre os DatasFrames.



(abaixo DataFrames para a prática do exposto acima)

Execute

```
import pandas as pd
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']},
                     index=[0, 1, 2, 3])

df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                    'B': ['B4', 'B5', 'B6', 'B7'],
                    'C': ['C4', 'C5', 'C6', 'C7'],
                    'D': ['D4', 'D5', 'D6', 'D7']},
                   index=[4, 5, 6, 7])

df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
                    'B': ['B8', 'B9', 'B10', 'B11'],
                    'C': ['C8', 'C9', 'C10', 'C11'],
                    'D': ['D8', 'D9', 'D10', 'D11']},
                   index=[8, 9, 10, 11])

df4 = pd.DataFrame({'B': ['B2', 'B3', 'B6', 'B7'],
                    'D': ['D2', 'D3', 'D6', 'D7'],
                    'F': ['F2', 'F3', 'F6', 'F7']},
                   index=[2, 3, 6, 7])

pd.concat([df1,df2,df4], axis=0)
```

7.2 *Status* Atual do Nosso Código (SIMCOSTA)

Nesse momento, o nosso programa deve estar como apresentado no código abaixo. Observe que:

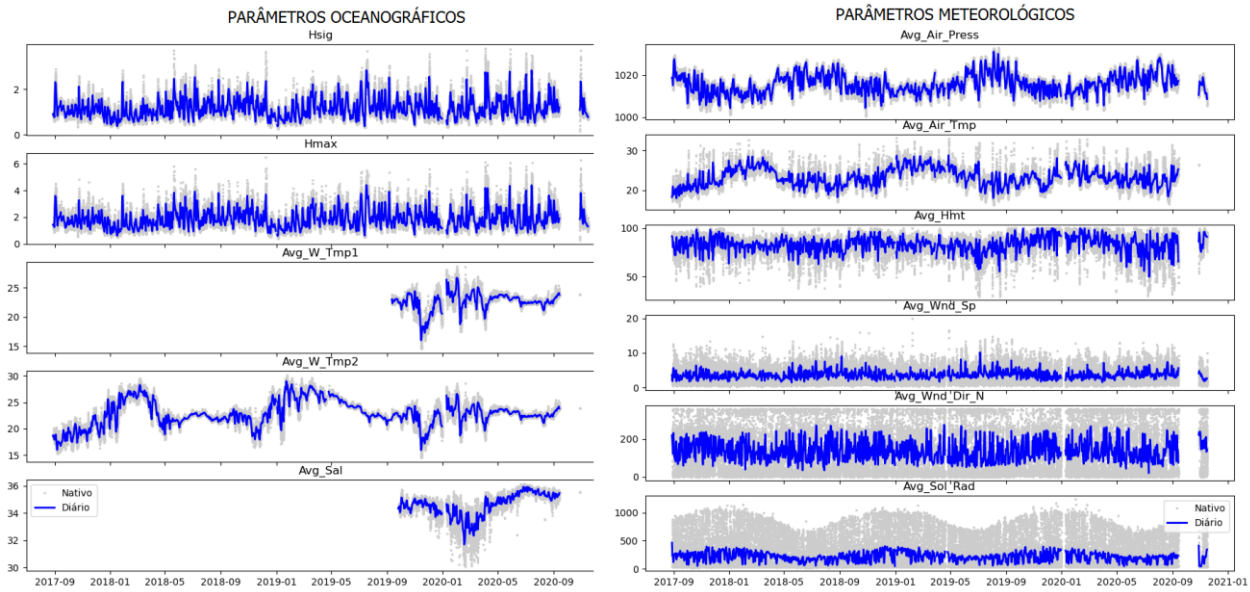
- dessa vez vamos trabalhar com a boia **RJ-4** (**Boia Axys**)
- utilizaremos os parâmetros **Oceanográficos** e **Meteorológicos** contidos nos arquivos **.csv**
  SIMCOSTA_RJ-4_**OCEAN**_2017-08-28_2020-11-16.**csv**
  SIMCOSTA_RJ-4_**MET**_2017-08-28_2020-11-16.**csv**

- não removeremos as linhas que estão marcadas com **nan** (deixaremos o Pandas lidar com essa condição).
- Os dados plotados tiveram reamostragem **diária**.

Execute

```
import pandas as pd
import numpy as np
import copy
import matplotlib.pyplot as plt

diretorio = 'C:/Negri/UERJ-AULAs/UERJ-2013/UERJ -aulas-2013/Aulas-MatematicaEspecial/TURMA 2020.1/'
mydict = dict() #dicionario vazio

'''================ ACESSO PARAMETROS OCEANOGRÁFICOS ================'''
nome_file = 'SIMCOSTA_RJ-4_OCEAN_2017-08-28_2020-11-16.csv'
df = pd.read_csv(diretorio+nome_file, header=34, sep=',')
lista = ['YEAR', 'MONTH', 'DAY', 'HOUR', 'MINUTE', 'SECOND',
         'Hsig','Hmax','Avg_W_Tmp1','Avg_W_Tmp2','Avg_Sal']
dfocean = copy.deepcopy(df[lista]) # subset DataFrame
dfocean['Avg_Sal'].values[dfocean['Avg_Sal'] < 30.0] = np.nan
dfocean.interpolate(method='linear', limit=7, inplace=True) # interpola
# recorde das colunas
df_recorte = dfocean.loc[:,['YEAR','MONTH','DAY','HOUR','MINUTE','SECOND']]
# indice formato timestamp (datetime)
dfocean.index = pd.to_datetime(df_recorte)
# remover as colunas
dfocean.drop(['YEAR','MONTH','DAY','HOUR','MINUTE','SECOND'],
             axis=1, inplace=True)
mydict['dfocean'] = dfocean.copy() # armazero no dicionario
'''================ FIM - ACESSO PARAMETROS OCEANOGRÁFICOS ================'''



'''================ ACESSO PARAMETROS METEOROLOGICOS ================'''
nome_file = 'SIMCOSTA_RJ-4_MET_2017-08-28_2020-11-16.csv'
df = pd.read_csv(diretorio+nome_file, header=21, sep=',')
lista = ['YEAR', 'MONTH', 'DAY', 'HOUR', 'MINUTE', 'SECOND',
         'Avg_Air_Press','Avg_Air_Tmp','Avg_Hmt','Avg_Wnd_Sp',
         'Avg_Wnd_Dir_N','Avg_Sol_Rad']
dfmeteo = copy.deepcopy(df[lista]) # subset DataFrame
# verificar se há outras pendências, por exemplo, 'None'
print (dfmeteo[dfmeteo.values == 'None'])
dfmeteo['Avg_Wnd_Dir_N'].replace('None', np.nan, inplace=True)

print (dfmeteo.dtypes) # verifica os tipos dos dados
#transforma para float
dfmeteo['Avg_Wnd_Dir_N']=dfmeteo['Avg_Wnd_Dir_N'].astype(float)

dfmeteo.interpolate(method='linear', limit=7, inplace=True) # interpola
```

```
# recorde das colunas
df_recorte = dfmeteo.loc[:,['YEAR','MONTH','DAY','HOUR','MINUTE','SECOND']]
# indice formato timestamp (datetime)
dfmeteo.index = pd.to_datetime(df_recorte)
# remover as colunas
dfmeteo.drop(['YEAR','MONTH','DAY','HOUR','MINUTE','SECOND'],
             axis=1, inplace=True)
mydict['dfmeteo'] = dfmeteo.copy() # armazero o dataframe no dicionario
'''================ FIM - ACESSO PARAMETROS METEOROLOGICOS ================'''

# plotagem dos dataframes: dfocean e dfmeteo
for i in mydict.keys():
    df = mydict[i]

    nlinhas, ncolunas = 5, 1
    if i == 'dfmeteo': nlinhas, ncolunas = 6, 1
    fig, axes = plt.subplots(nlinhas, ncolunas, figsize=(11, 10), sharex=True)

    parametros = df.columns
    tit_graficos = df.columns
    for param, titulo, ax in zip(parametros,tit_graficos,axes):
        ax.plot(df[param], marker='.', markersize=3, color='0.8',
                linestyle='None', label='Nativo')
        ax.plot(df[param].resample(rule='D').mean(), color='blue',
                linewidth=2, label='Diário')
        ax.set_title(titulo)
    plt.legend()
```



Como é possível observar nas figuras acima, temos os dados oceanográficos e meteorológicos. Nosso objetivo agora é concatenar esses dois

conjuntos de dados em um único DataFrame e realizar as operações estatísticas e plotar as informações.

Para **concatenação** dos DataFrames (*dfocean* e *dfmeteo*) faremos uso do método **df.concat()** visto anteriormente.

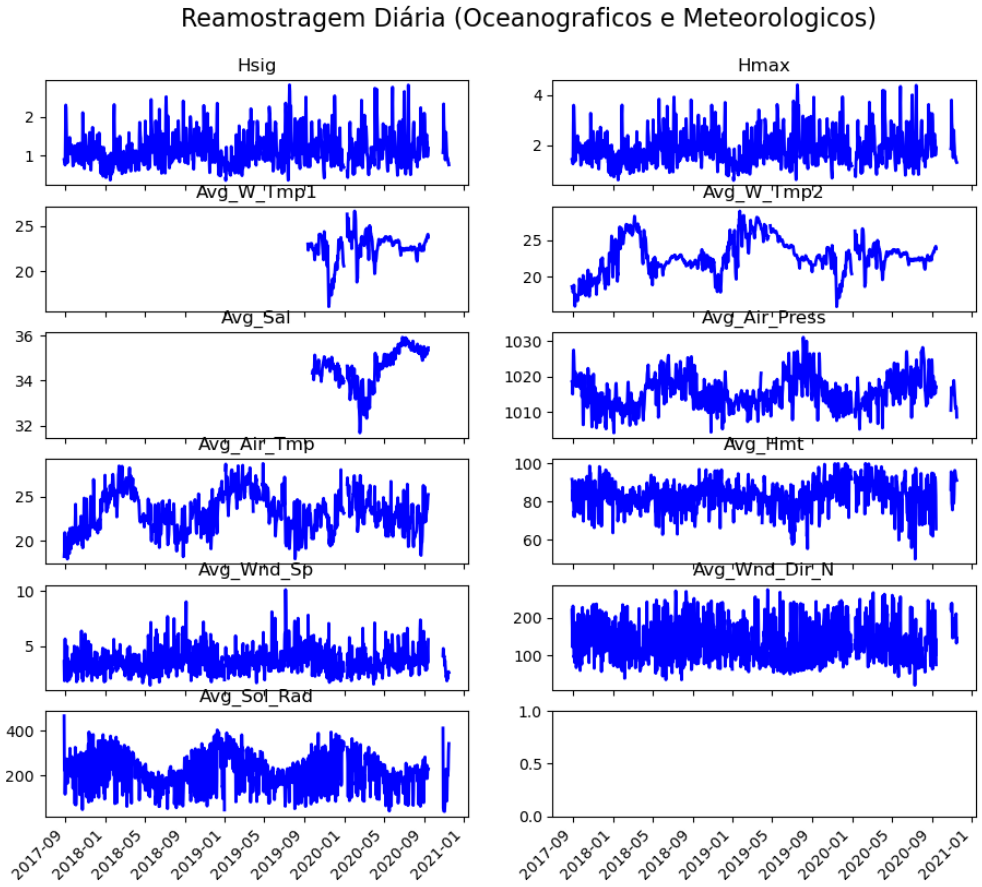
Observe e execute o código abaixo.

> **Atenção** : o código abaixo é continuação do código acima, portanto, é necessário primeiro executar o código acima para depois continuar com o código abaixo.

Execute

```
'''================ CONCATENAÇÃO DOS DATAS FRAMES ================'''
# OBS: como alterar a visualização dos data frames no console
pd.set_option('max_columns', None) # para mostrar todas as colunas do DataFrame
pd.set_option('max_columns', 10) # para mostrar 10 colunas
pd.reset_option('max_columns') # para voltar a posição default

# primeiro um teste, para visualizar o resultado da concatenação
dfocean_s = dfocean.iloc[0:200,0:2].resample(rule='D').mean()
dfmeteo_s = dfmeteo.iloc[0:200,0:2].resample(rule='D').mean()
# preserva as colunas e combina os indices
df = pd.concat([dfocean_s, dfmeteo_s], axis=1)
print (df)

# agora é pra valer!!!
dfocean_s = dfocean.resample(rule='D').mean()
dfmeteo_s = dfmeteo.resample(rule='D').mean()
df = pd.concat([dfocean_s, dfmeteo_s], axis=1)
print (df.head)

# plotagem do dataframe resultante da concatenação
fig, axes = plt.subplots(6, 2, figsize=(11, 10), sharex=True)
axes = np.ravel(axes) # lineariza os array com objetos eixos
parametros = df.columns
tit_graficos = df.columns
for param, titulo, ax in zip(parametros,tit_graficos,axes):
    ax.plot(df[param], color='blue', linewidth=2)
    ax.set_title(titulo)

# rotaciona o label de todos os eixos da figura
# https://stackoverflow.com/questions/10998621/rotate-axis-text-in-python-matplotlib
fig.autofmt_xdate(rotation=45)
fig.suptitle('Reamostragem Diária', fontsize=16)

'''================ fim - CONCATENAÇÃO DOS DATAS FRAMES ================'''
```

O gráfico abaixo evidencia os parâmetros meteorológicos e oceanográficos, reamostrados com frequência temporal diária, que compõem o dataframe concatenado '*df*'.



Agora que temos um único DataFrame com os parâmetros oceanográficos e meteorológicos. Vamos utilizar a biblioteca **Seaborn** para visualizar as informações.

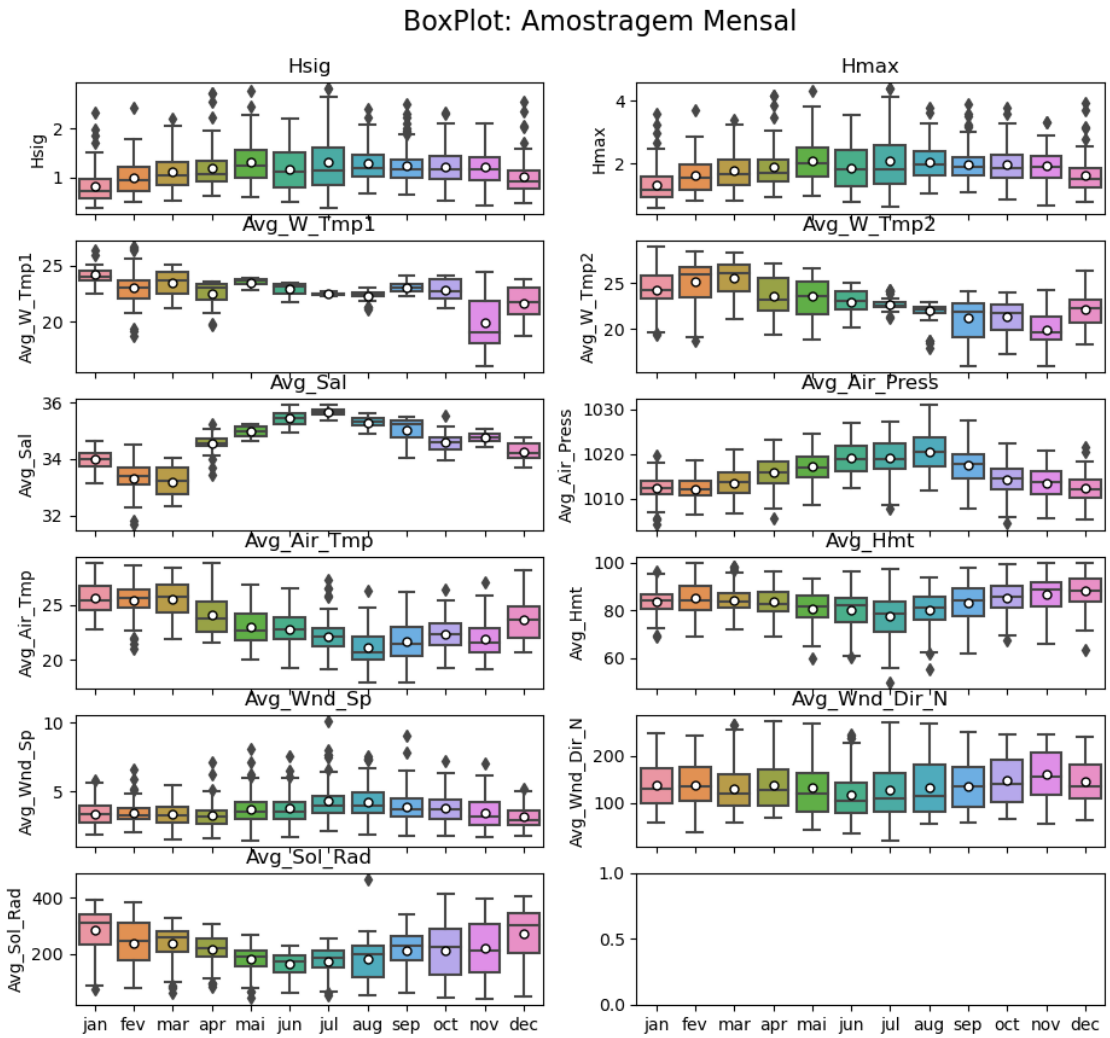
O **Seaborn** é uma biblioteca de visualização que fica apoiada na biblioteca mais básica **Matplotlib**, e pessui funções gráficas eficientes que facilitam a elaboração de figuras e gráficos.

No código abaixo temos uma plotagem do tipo **boxplot** (https://seaborn.pydata.org/generated/seaborn.boxplot.html).

Execute

```
'''================ Plotagens Seaborn: boxplot ================'''
import seaborn as sns
# plotagem do dataframe resultante da concatenação
fig, axes = plt.subplots(6, 2, figsize=(11, 10), sharex=True)
```

```
axes = np.ravel(axes) # lineariza os array com objetos eixos
parametros = df.columns

meanprops={"marker":"o",
           "markerfacecolor":"white",
           "markeredgecolor":"black",
           "markersize":"5"} # como será o simbolo que define a media (mean)

for param, ax in zip(parametros,axes):
    sns.boxplot(data=df, x=df.index.month, showmeans=True,
                meanprops=meanprops, y=param, ax=ax)
    ax.set_ylabel(param)
    ax.set_title(param)

fig.suptitle('BoxPlot: Amostragem Mensal', fontsize=16)

# posso mudar os labels dos eixos X
ax.set_xticklabels(['jan','fev','mar','apr','mai',
                    'jun','jul','aug','sep','oct','nov','dec'])
'''================ fim - Plotagens Seaborn: boxplot ================'''
```



No código abaixo, primeiramente utilizamos o método `.corr` para calcular a correlação cruzada dos parâmetros meteorológicos e oceanográficos presentes no DataFrame.

Em seguida, foi gerado o gráfico **heatmap** do **Seaborn**: (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

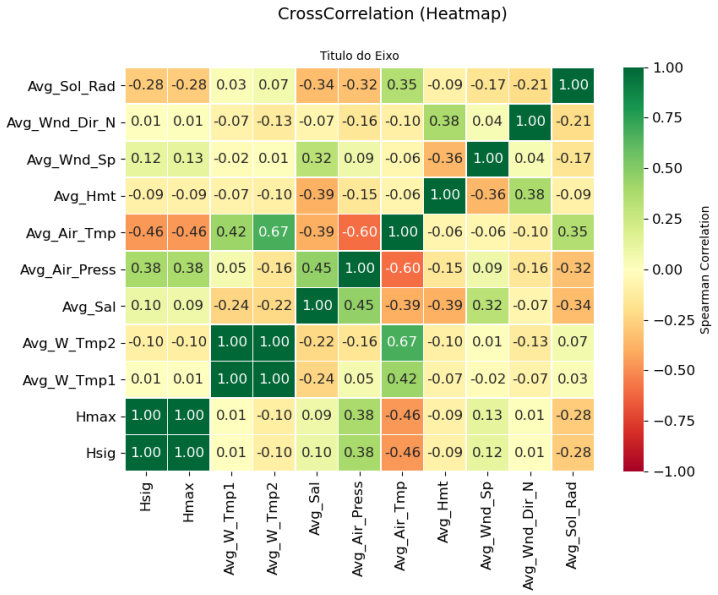
Observe que *cmap* é uma das *keywords* possíveis de serem utilizadas no comando *sns.heatmap*. Essa keyword permite que façamos escolha **da tabela de cores** que vamos utilizar para plotagem do **heatmap**. O **Seaborn** suporta diversas tabelas de cores:

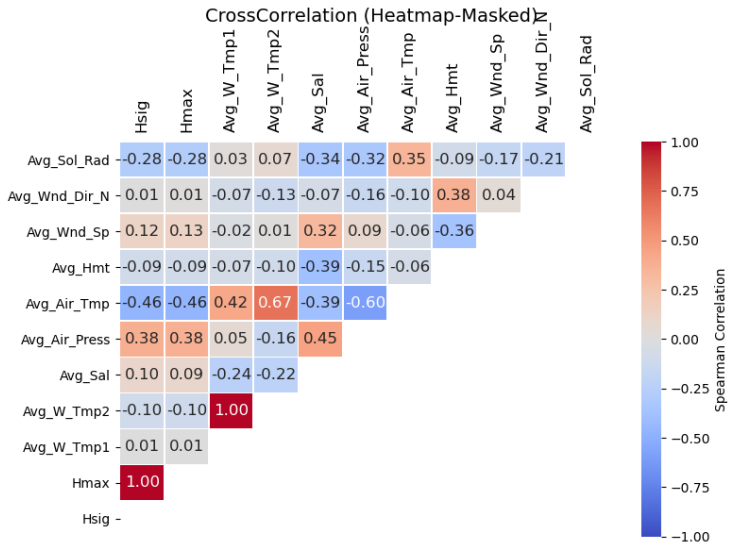
https://seaborn.pydata.org/tutorial/color_palettes.html
https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html

Execute

```
'''================ Plotagens Seaborn: Heatmap (Correlação) ================'''
# correlação entre as colunas (crosscorrelation)
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html
df_correl = df.corr(method='spearman')

fig, ax = plt.subplots(figsize=(8,6))
#https://seaborn.pydata.org/generated/seaborn.heatmap.html
shm = sns.heatmap(df_correl, cmap='RdYlGn',  # cmpa é referente à tabela de cores (colormap))
                  linewidths=0.5, annot=True, ax=ax,
                  vmin = -1, vmax = 1, center = 0, fmt='.2f',
                  annot_kws={'size':12},
                  cbar_kws={'label': 'Spearman Correlation'})
# poderia mudar a tabela de cores do heatmap para 'cmap='Spectral'

shm.set_xticklabels(shm.get_xticklabels(), rotation = 90, fontsize = 12)
shm.set_yticklabels(shm.get_yticklabels(), rotation = 0, fontsize = 12)

cbar = ax.collections[0].colorbar # seleciona a tabela de cores
cbar.ax.tick_params(labelsize=12) # altera o size dos labels numericos da colorbar

ax.set_title('Titulo do Eixo', fontsize=10)
fig.suptitle('CrossCorrelation (Heatmap) - Titulo da Figura', fontsize=14)
ax.invert_yaxis() # inverte o eixo Y

# posso mudar os labels dos eixos X e Y, caso necessário
ax.set_xticklabels(np.arange(11))
ax.set_yticklabels(np.arange(11))
'''================ fim - Plotagens Seaborn: Heatmap (Correlação) ================'''



'''================ Plotagens Seaborn: Heatmap Masked (Correlação) ================'''
# construindo uma máscara (nova plotagem)
fig1, ax1 = plt.subplots(figsize=(8,6))
mask = np.zeros(df_correl.shape, dtype=bool)
mask[np.triu_indices(len(mask))] = True
shm1 = sns.heatmap(df_correl, cmap = 'coolwarm',
                   linewidths=0.5, annot=True, ax=ax1,
```

```
                    vmin = -1, vmax = 1, center = 0,  fmt='0.2f',
                    annot_kws={'size':12},
                    cbar_kws={'label': 'Spearman Correlation'},
                    mask = mask)
ax1.invert_yaxis() # inverte o eixo Y
plt.tick_params(axis='both', which='major', labelbottom = False, bottom=False,
top = False, labeltop=True)
shm1.set_xticklabels(shm1.get_xticklabels(), rotation = 90, fontsize = 12)
fig1.suptitle('CrossCorrelation (Heatmap-Masked)', fontsize=14)
plt.tight_layout() # fixa apropriadamente a diagramação da figura
#plt.savefig(indir1+'heatmap_correl_', dpi = 300, bbox_inches='tight')
'''================ fim Plotagens Seaborn: Heatmap Masked (Correlação) ================'''
```





Outra plotagem interessante do Heatmap com anotações pode ser encontrada nesse *site*: https://seaborn.pydata.org/examples/spreadsheet_heatmap.html

Uma função bastante interessante do Seaborn é a **.pairplot** (https://seaborn.pydata.org/generated/seaborn.pairplot.html). A invocação mais simples usa o gráfico de dispersão *scatterplot()* para cada par de variáveis, e gráfico de histograma *histplot()* para os gráficos marginais ao longo da diagonal.

Execute

```
'''================ Plotagens Seaborn: PairPlot ================'''
# simples plotagem (média semanal)
lista = ['Hsig', 'Avg_W_Tmp2', 'Avg_Sal', 'Avg_Wnd_Sp']
sns.pairplot(df[lista].resample(rule='W').mean()) # reamostragem semanal
plt.tight_layout()

# separado por mês (média semanal)
dfx = df[lista].resample(rule='W').mean() # reamostragem semanal
dfx['meses'] = dfx.index.month #vamos criar uma coluna com os meses
sns.pairplot(dfx, hue="meses", palette='gist_ncar') #palette='nipy_spectral'

# terceiro (dados diários, porém plota mês a mês)
lista = ['Hsig', 'Avg_W_Tmp2', 'Avg_Sal', 'Avg_Wnd_Sp']
for i in np.arange(1,13): # array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
    dfx = df[df.index.month == i]
    sns.pairplot(dfx[lista])
plt.tight_layout()
'''================ fim - Plotagens Seaborn: PairPlot ================'''
```