

SCANPLOT-multiple

June 16, 2020

1 SCANPLOT - Um sistema de plotagem simples para o SCANTEC

O SCANPLOT é um módulo escrito em linguagem Python preparado para ler e plotar as tabelas com as estatísticas do Sistema Comunitário de Avaliação de modelos Numéricos de Tempo e Clima (SCANTEC*). O seu uso pode ser feito por meio da linha de comando ou através do Jupyter. O SCANPLOT transforma as tabelas do SCANTEC em dataframes do Pandas e pode ser facilmente estendido a partir da introdução de funções para a plotagem destes dataframes na forma como o usuário precisar.

O módulo `scanplot` possui as seguintes funções:

1. `read_namelist`: esta função lê os arquivos de namelist e definições dos modelos do SCANTEC;
2. `get_dataframe`: esta função transforma uma ou mais tabelas em dataframes do Pandas, acessíveis por meio de um dicionário;
3. `plot_lines`: esta função plota gráficos de linhas a partir dos dataframes.

As funções possuem formas específicas de utilização. Para saber como utilizá-las, carregue primeiro a função a partir do módulo principal (por exemplo, a função `read_namelist`):

```
from scanplot import read_namelist
```

E em seguida, digite uma das suas formas a seguir:

```
help(read_namelist)
```

ou

```
print(read_namelist.__doc__)
```

*de [MATTOS, J. G. Z.](#); [SAPUCCI, L. F.](#). **SCANTEC - SISTEMA COMUNITÁRIO DE AVALIAÇÃO DE MODELOS NUMÉRICOS DE TEMPO E CLIMA. 2017.** Patente: Programa de Computador. Número do registro: BR512017000576-1, data de registro: 30/01/2017, Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial.

1.1 Leitura dos namelists do SCANTEC

O SCANTEC é um software de linha de comando escrito em linguagem Fortran preparado para ler, interpolar e calcular as estatísticas básicas (Viés, Raiz do Erro Quadrático Médio e Correlação de Anomalias) a partir de modelos de previsão numérica de tempo, como os modelos BAM, BRAMS e

Eta. O SCANPLOT faz o trabalho de plotar os resultados a partir das tabelas com o resumo destas estatísticas. Para utilizar o SCANPLOT, o usuário deve ler os arquivos de namelist e definições dos modelos utilizados nas avaliações de forma que o software saiba quais foram as definições utilizadas pelo usuário e em que local estão armazenadas as tabelas com os resultados.

Para isso, basta carregar a função `read_namelist`s a partir do módulo principal `scanplot`, com o seguinte comando:

```
[1]: from scanplot import read_namelist
```

Para conhecer como deve ser utilizada a função `read_namelist`s, o usuário pode utilizar um dos comandos a seguir:

```
[2]: print(read_namelist.__doc__)  
      #help(read_namelist)
```

```
read_namelist  
=====
```

Esta função lê os namelist e arquivos de definições dos modelos do SCANTEC e
retorna para o usuário dois dicionários, `VarsLevs` e `Confs`, com as informações lidas.

Parâmetros de entrada

`basepath` : diretório raiz da instalação do SCANTEC.

Resultados

`VarsLevs` : dicionário com as variáveis, níveis e nomes definidos no
arquivo `scantec.vars`
 `Confs` : dicionário com as definições contidas no arquivo `scantec.conf`

Uso

```
from scanplot import read_namelist  
data_vars, data_conf = read_namelist("~/SCANTEC")
```

A função `read_namelist`s recebe um caminho (raiz da instalação do SCANTEC) como parâmetro de entrada e retorna para o usuário dois dicionários, os quais contém as informações dos arquivos `scantec.conf` e `scantec.vars` do SCANTEC. Estes arquivos possuem as definições dos modelos (intervalo de tempo da avaliação, nome do modelo, resolução, caminhos etc). Os nomes `data_vars` e `data_conf` são os nomes dos objetos que serão criados e que conterão os dicionários com as definições dos arquivos `scantec.vars` e `scantec.conf`, respectivamente. A escolha destes nomes fica a critério do usuário.

```
[3]: data_vars, data_conf = read_namelists("/Volumes/RAID0/carlos/Documents/INPE2020/
↳SCANTEC/ilopolis/SCANTEC.2.0.0b1")
```

Para inspecionar o conteúdo e a estrutura dos dados contidos nos objetos `data_conf` e `data_vars`, basta digitar os nomes no prompt:

```
[4]: data_conf
```

```
[4]: {'Starting Time': datetime.datetime(2014, 8, 5, 0, 0),
      'Ending Time': datetime.datetime(2014, 8, 6, 0, 0),
      'Analysis Time Step': '12',
      'Forecast Time Step': '24',
      'History Time': '48',
      'scantec tables': '/home/carlos.bastarz/SCANTEC.2.0.0b1/tables',
      'run domain number': '1',
      'run domain lower left lat': '-49.875',
      'run domain lower left lon': '-82.625',
      'run domain upper right lat': '11.375',
      'run domain upper right lon': '-35.375',
      'run domain resolution dx': '0.4',
      'run domain resolution dy': '0.4',
      'Reference Model Name': 'BAM_TQ0299L064_1',
      'Reference file': '/dados/das/public/SCANTEC/TestCase/AGCM/TQ0299L064/%y4%m2%d2%h2/GPOSNMC%y4%m2%d2%h2%y4%m2%d2%h2P.icn.TQ0299L064.ctl',
      'Experiments': {'EXP01': ['BAM_TQ0299L064_1',
                                '/dados/das/public/SCANTEC/TestCase/AGCM/TQ0299L064/%y4%m2%d2%h2/GPOSNMC%iy4%im2%id2%ih2%fy4%fm2%fd2%fh2P.fct.TQ0299L064.ctl'],
                       'EXP02': ['BAM_TQ0299L064_1',
                                '/dados/das/public/SCANTEC/TestCase/AGCM/TQ0299L064/%y4%m2%d2%h2/GPOSNMC%iy4%im2%id2%ih2%fy4%fm2%fd2%fh2P.fct.TQ0299L064.ctl']}},
      'Climatology Model Name': '3',
      'Climatology file':
'/dados/das/public/SCANTEC/climatologia/climatologia50yr.%mc.bin',
      'Output directory': '/Volumes/RAID0/carlos/Documents/INPE2020/SCANTEC/ilopolis/SCANTEC.2.0.0b1/dataout'}
```

```
[5]: data_vars
```

```
[5]: {0: ('VTMP:925', 'Virtual Temperature @ 925 hPa [K]'),
      1: ('VTMP:850', 'Virtual Temperature @ 850 hPa [K]'),
      2: ('VTMP:500', 'Virtual Temperature @ 500 hPa [K]'),
      3: ('TEMP:850', 'Absolute Temperature @ 850 hPa [K]'),
      4: ('TEMP:500', 'Absolute Temperature @ 500 hPa [K]'),
      5: ('TEMP:250', 'Absolute Temperature @ 250 hPa [K]'),
      6: ('PSNM:000', 'Pressure reduced to MSL [hPa]'),
      7: ('UMES:925', 'Specific Humidity @ 925 hPa [g/Kg]'),
      8: ('UMES:850', 'Specific Humidity @ 850 hPa [g/Kg])'}
```

```

9: ('UMES:500', 'Specific Humidity @ 500 hPa [g/Kg]'),
10: ('AGPL:925', 'Inst. Precipitable Water @ 925 hPa [Kg/m2]'),
11: ('ZGEO:850', 'Geopotential height @ 850 hPa [gpm]'),
12: ('ZGEO:500', 'Geopotential height @ 500 hPa [gpm]'),
13: ('ZGEO:250', 'Geopotential height @ 250 hPa [gpm]'),
14: ('UVEL:850', 'Zonal Wind @ 850 hPa [m/s]'),
15: ('UVEL:500', 'Zonal Wind @ 500 hPa [m/s]'),
16: ('UVEL:250', 'Zonal Wind @ 250 hPa [m/s]'),
17: ('VVEL:850', 'Meridional Wind @ 850 hPa [m/s]'),
18: ('VVEL:500', 'Meridional Wind @ 500 hPa [m/s]'),
19: ('VVEL:250', 'Meridional Wind @ 250 hPa [m/s]')}

```

Com as informações dos arquivos de namelist do SCANTEC carregados, o próximo passo é ler as tabelas geradas na avaliação com o SCANTEC e transformá-las em dataframes do Pandas. Para isso, o usuário deverá utilizar a função `get_dataframe` do módulo `scanplot`:

```
[6]: from scanplot import get_dataframe
```

Da mesma forma como foi feito com a função `read_namelists`, pode-se digitar o comando `print(funcao.__doc__)` ou simplesmente, `help(funcao)` para descobrir como a função deve ser utilizada:

```
[7]: #print(get_dataframe.__doc__)
help(get_dataframe)
```

Help on function `get_dataframe` in module `scanplot`:

```

get_dataframe(dataInicial, dataFinal, Stats, Exps, outDir)
get_dataframe
=====

```

Esta função transforma a(s) tabela(s) do SCANTEC em dataframe(s).

Parâmetros de entrada

```

dataInicial : objeto datetime com a data inicial do experimento
dataFinal   : objeto datetime com a data final do experimento
Stats       : lista com os nomes das estatísticas a serem processadas
Exps        : lista com os nomes dos experimentos
outDir      : string com o diretório com as tabelas do SCANTEC

```

Resultado

Dicionário com o(s) dataframe(s) com a(s) tabela(s) do SCANTEC.

Uso

```
from scanplot import get_dataframe
```

```

dataInicial = data_conf["Starting Time"]
dataFinal = data_conf["Ending Time"]
Stats = ["ACOR", "RMSE", "VIES"]
Exps = list(data_conf["Experiments"].keys())
outDir = data_conf["Output directory"]

dTable = get_dataframe(dataInicial,dataFinal,Stats,Exps,outDir)

```

A função `get_dataframe` recebe uma série de parâmetros de entrada e retorna um dicionário com uma ou mais tabelas que já estarão no formato de dataframe do Pandas. Na célula a seguir, serão definidos os valores de entrada da função `get_dataframe` a partir dos dicionários `data_conf` e `data_vars`, criados anteriormente.

Observe que os parâmetros `Vars` e `Stats` são atribuídos de formas diferentes dos demais. O parâmetro `Stats` é uma lista que deve possuir pelo menos um elemento e ele sempre deve possuir a forma `Stat = [...]`. Na versão atual do SCANPLOT, o usuário pode escolher as estatísticas ACOR (correlação de anomalias), RMSE (raiz do erro quadrático médio) e VIES (viés), em qualquer ordem ou combinação entre elas. O parâmetro `Vars` também é uma lista, mas é definido de forma diferente. O usuário deve observar que no dicionário `data_vars`, para cada índice está associado uma tupla do tipo `('VAR:LEV', 'Nome da Variável @ Nível hPa [unidade]')`. Isto foi feito para facilitar ao usuário a escolha da variável, pois ao invés de se digitar o nome da variável, basta escolher pelo menos um dos índices do dicionário `data_vars` que deseja, da seguinte forma `Vars = list(map(data_vars.get, [1,2,3,...]))`.

```

[8]: dataInicial = data_conf["Starting Time"]
dataFinal = data_conf["Ending Time"]
Vars = list(map(data_vars.get, [12,14]))
Stats = ["ACOR", "RMSE", "VIES"]
Exps = list(data_conf["Experiments"].keys())
outDir = data_conf["Output directory"]

```

Com a definição dos parâmetros de entrada da função `get_dataframe`, a sua utilização é feita da seguinte forma:

```

[9]: dTable = get_dataframe(dataInicial,dataFinal,Stats,Exps,outDir)

```

Na chamada da função `get_dataframe`, o objeto `dTable` é um dicionário que deverá conter as tabelas escolhidas pelo usuário a partir do ajuste dos parâmetros de entrada da função. Para inspecionar o conteúdo do dicionário `dTable`, basta digitar no prompt:

```

[10]: dTable

```

```

[10]: {'ACOREXP01_20140805002014080600T.scan':      %Previsao  vtmp:925  vtmp:850
vtmp:500  temp:850  temp:500  temp:250  \
0          0        0.0      0.0      0.0      1.000      1.000      1.000
1         24        0.0      0.0      0.0      0.993      0.997      0.992
2         48        0.0      0.0      0.0      0.986      0.984      0.981

```

3	72	0.0	0.0	0.0	0.975	0.961	0.972
---	----	-----	-----	-----	-------	-------	-------

	psnm:000	umes:925	umes:850	...	agpl:925	zgeo:850	zgeo:500	zgeo:250	\
0	1.000	1.000	1.000	...	1.000	1.000	1.000	1.000	
1	0.982	0.975	0.966	...	0.984	0.991	0.999	0.999	
2	0.970	0.942	0.931	...	0.963	0.981	0.993	0.993	
3	0.951	0.943	0.908	...	0.943	0.976	0.984	0.984	

	uvel:850	uvel:500	uvel:250	vvel:850	vvel:500	vvel:250
0	1.000	1.000	1.000	1.000	1.000	1.000
1	0.972	0.986	0.987	0.939	0.965	0.969
2	0.959	0.978	0.968	0.892	0.869	0.853
3	0.941	0.965	0.930	0.821	0.824	0.766

[4 rows x 21 columns],

	'ACOREXP02_20140805002014080600T.scan':				%Previsao	vtmp:925	vtmp:850
	vtmp:500	temp:850	temp:500	temp:250	\		
0	0	1.0	1.0	1.0	1.000	1.000	1.000
1	24	0.9	0.9	0.9	0.993	0.997	0.992
2	48	0.8	0.8	0.8	0.986	0.984	0.981
3	72	0.7	0.7	0.7	0.975	0.961	0.972

	psnm:000	umes:925	umes:850	...	agpl:925	zgeo:850	zgeo:500	zgeo:250	\
0	1.000	1.000	1.000	...	1.000	1.000	1.000	1.000	
1	0.982	0.975	0.966	...	0.984	0.991	0.999	0.999	
2	0.970	0.942	0.931	...	0.963	0.981	0.993	0.993	
3	0.951	0.943	0.908	...	0.943	0.976	0.984	0.984	

	uvel:850	uvel:500	uvel:250	vvel:850	vvel:500	vvel:250
0	1.000	1.000	1.000	1.000	1.000	1.000
1	0.972	0.986	0.987	0.939	0.965	0.969
2	0.959	0.978	0.968	0.892	0.869	0.853
3	0.941	0.965	0.930	0.821	0.824	0.766

[4 rows x 21 columns],

	'RMSEEXP01_20140805002014080600T.scan':				%Previsao	vtmp:925	vtmp:850
	vtmp:500	temp:850	temp:500	temp:250	\		
0	0	0.0	0.0	0.0	0.000	0.000	0.000
1	24	0.0	0.0	0.0	1.216	0.738	0.623
2	48	0.0	0.0	0.0	1.830	1.576	1.207
3	72	0.0	0.0	0.0	2.392	2.487	1.746

	psnm:000	umes:925	umes:850	...	agpl:925	zgeo:850	zgeo:500	zgeo:250	\
0	0.000	0.000	0.000	...	0.000	0.000	0.000	0.000	
1	1.425	0.001	0.001	...	3.164	9.861	13.228	13.228	
2	2.264	0.002	0.002	...	4.816	15.122	25.499	25.499	
3	2.630	0.002	0.002	...	6.028	17.548	39.672	39.672	

	uvel:850	uvel:500	uvel:250	vvel:850	vvel:500	vvel:250
0	0.000	0.000	0.000	0.000	0.000	0.000
1	2.300	2.586	3.744	2.353	2.481	4.219
2	2.794	3.291	5.608	2.841	3.713	7.185
3	3.392	4.179	8.191	3.149	4.558	9.087

[4 rows x 21 columns],

	'RMSEEXP02_20140805002014080600T.scan':				%Previsao	vtmp:925	vtmp:850
	vtmp:500	temp:850	temp:500	temp:250	\		
0	0	0.1	0.1	0.1	0.000	0.000	0.000
1	24	0.2	0.2	0.2	1.216	0.738	0.623
2	48	0.3	0.3	0.3	1.830	1.576	1.207
3	72	0.4	0.4	0.4	2.392	2.487	1.746

	psnm:000	umes:925	umes:850	...	agpl:925	zgeo:850	zgeo:500	zgeo:250	\
0	0.000	0.000	0.000	...	0.000	0.000	0.000	0.000	
1	1.425	0.001	0.001	...	3.164	9.861	13.228	13.228	
2	2.264	0.002	0.002	...	4.816	15.122	25.499	25.499	
3	2.630	0.002	0.002	...	6.028	17.548	39.672	39.672	

	uvel:850	uvel:500	uvel:250	vvel:850	vvel:500	vvel:250
0	0.000	0.000	0.000	0.000	0.000	0.000
1	2.300	2.586	3.744	2.353	2.481	4.219
2	2.794	3.291	5.608	2.841	3.713	7.185
3	3.392	4.179	8.191	3.149	4.558	9.087

[4 rows x 21 columns],

	'VIESEXP01_20140805002014080600T.scan':				%Previsao	vtmp:925	vtmp:850
	vtmp:500	temp:850	temp:500	temp:250	\		
0	0	0.0	0.0	0.0	0.000	0.000	0.000
1	24	0.0	0.0	0.0	-0.670	-0.279	0.118
2	48	0.0	0.0	0.0	-1.113	-0.687	-0.239
3	72	0.0	0.0	0.0	-1.503	-1.331	-0.926

	psnm:000	umes:925	umes:850	...	agpl:925	zgeo:850	zgeo:500	zgeo:250	\
0	0.000	0.0	0.000	...	0.000	0.000	0.000	0.000	
1	0.411	-0.0	-0.000	...	-1.228	0.643	-6.138	-6.138	
2	1.248	-0.0	-0.000	...	-2.063	5.772	-7.402	-7.402	
3	1.216	-0.0	-0.001	...	-2.606	3.845	-16.953	-16.953	

	uvel:850	uvel:500	uvel:250	vvel:850	vvel:500	vvel:250
0	0.000	0.000	0.000	0.000	0.000	0.000
1	-0.208	0.406	-0.528	-0.382	0.076	-0.157
2	0.062	0.910	0.006	-0.439	-0.004	-0.050
3	0.456	1.522	1.223	-0.464	-0.153	-0.660

```
[4 rows x 21 columns],
'VIESEXP02_20140805002014080600T.scan':    %Previsao  vtmp:925  vtmp:850
vtmp:500  temp:850  temp:500  temp:250  \
0         0        0.1      0.1      0.1      0.000      0.000      0.000
1        24        0.2      0.2      0.2     -0.670     -0.279      0.118
2        48        0.3      0.3      0.3     -1.113     -0.687     -0.239
3        72        0.4      0.4      0.4     -1.503     -1.331     -0.926

      psnm:000  umes:925  umes:850  ...  agpl:925  zgeo:850  zgeo:500  zgeo:250  \
0      0.000      0.0      0.000  ...    0.000      0.000      0.000      0.000
1      0.411     -0.0     -0.000  ...   -1.228      0.643     -6.138     -6.138
2      1.248     -0.0     -0.000  ...   -2.063      5.772     -7.402     -7.402
3      1.216     -0.0     -0.001  ...   -2.606      3.845    -16.953    -16.953

      uvel:850  uvel:500  uvel:250  vvel:850  vvel:500  vvel:250
0      0.000      0.000      0.000      0.000      0.000      0.000
1     -0.208      0.406     -0.528     -0.382      0.076     -0.157
2      0.062      0.910      0.006     -0.439     -0.004     -0.050
3      0.456      1.522      1.223     -0.464     -0.153     -0.660

[4 rows x 21 columns]}
```

No dicionário dTable, observe que foram carregadas as tabelas referente às estatísticas escolhidas (VIES, RMS e ACOR). Para visualizar o dataframe da tabela, basta passar o nome da tabela como key do dicionário dTable, como em dTable['NOME_TABELA']. Veja o exemplo a seguir:

```
[11]: dTable['ACOREXP01_20140805002014080600T.scan']
```

```
[11]:    %Previsao  vtmp:925  vtmp:850  vtmp:500  temp:850  temp:500  temp:250  \
0         0        0.0      0.0      0.0      1.000      1.000      1.000
1        24        0.0      0.0      0.0      0.993      0.997      0.992
2        48        0.0      0.0      0.0      0.986      0.984      0.981
3        72        0.0      0.0      0.0      0.975      0.961      0.972

      psnm:000  umes:925  umes:850  ...  agpl:925  zgeo:850  zgeo:500  zgeo:250  \
0      1.000      1.000      1.000  ...    1.000      1.000      1.000      1.000
1      0.982      0.975      0.966  ...    0.984      0.991      0.999      0.999
2      0.970      0.942      0.931  ...    0.963      0.981      0.993      0.993
3      0.951      0.943      0.908  ...    0.943      0.976      0.984      0.984

      uvel:850  uvel:500  uvel:250  vvel:850  vvel:500  vvel:250
0      1.000      1.000      1.000      1.000      1.000      1.000
1      0.972      0.986      0.987      0.939      0.965      0.969
2      0.959      0.978      0.968      0.892      0.869      0.853
3      0.941      0.965      0.930      0.821      0.824      0.766

[4 rows x 21 columns]
```



```
[12]: dTable['ACOREXP02_20140805002014080600T.scan']
```

```
[12]: %Previsao  vtmp:925  vtmp:850  vtmp:500  temp:850  temp:500  temp:250  \
0          0      1.0      1.0      1.0      1.000  1.000  1.000
1         24      0.9      0.9      0.9      0.993  0.997  0.992
2         48      0.8      0.8      0.8      0.986  0.984  0.981
3         72      0.7      0.7      0.7      0.975  0.961  0.972

      psnm:000  umes:925  umes:850  ...  agpl:925  zgeo:850  zgeo:500  zgeo:250  \
0      1.000      1.000      1.000  ...      1.000      1.000      1.000      1.000
1      0.982      0.975      0.966  ...      0.984      0.991      0.999      0.999
2      0.970      0.942      0.931  ...      0.963      0.981      0.993      0.993
3      0.951      0.943      0.908  ...      0.943      0.976      0.984      0.984

      uvel:850  uvel:500  uvel:250  vvel:850  vvel:500  vvel:250
0      1.000      1.000      1.000      1.000      1.000      1.000
1      0.972      0.986      0.987      0.939      0.965      0.969
2      0.959      0.978      0.968      0.892      0.869      0.853
3      0.941      0.965      0.930      0.821      0.824      0.766
```

```
[4 rows x 21 columns]
```

1.2 Explorando os dataframes

Dataframes do Pandas são dados tabulados que possuem uma série de funções e métodos que podem ser aplicados também com as tabelas do SCANTEC. Veja nos exemplos a seguir forma de selecionar as colunas e plotar os dados de forma individual ou agrupada.

No exemplo a seguir, é feita a seleção da coluna referente à correção de anomalias da temperatura absoluta em 850 hPa. Para isso, utiliza-se o método `loc` para fazer a subseleção do dataframe. O método `loc` em um dataframe indexa os valores da tabela a partir dos índices da linha e da coluna, respectivamente (`loc[linha,coluna]`):

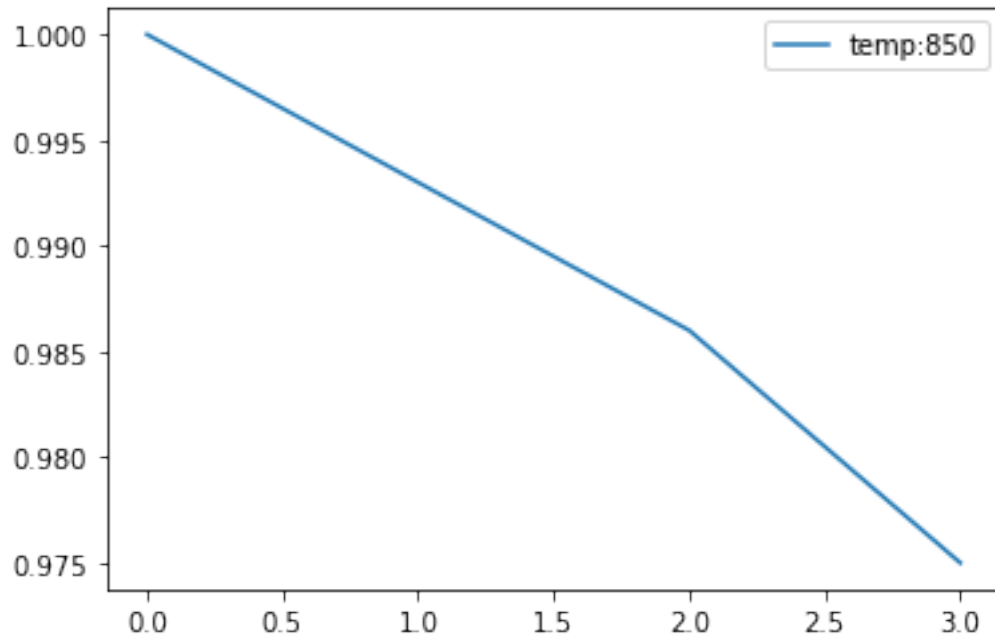
```
[13]: dTable['ACOREXP01_20140805002014080600T.scan'].loc[:, "temp:850"]
```

```
[13]: 0      1.000
      1      0.993
      2      0.986
      3      0.975
      Name: temp:850, dtype: float64
```

O dataframe do Pandas permite também realizar a plotagem da subseleção realizada utilizando a função `plot()`. Veja no exemplo a seguir:

```
[14]: dTable['ACOREXP01_20140805002014080600T.scan'].loc[:, ["temp:850"]].plot()
```

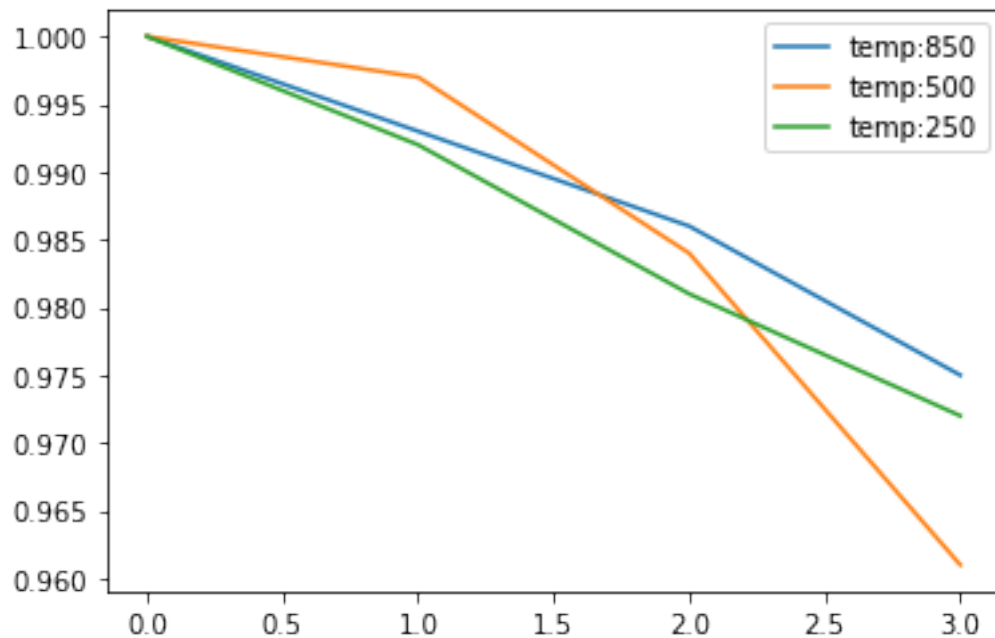
```
[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9061c410d0>
```



Com o método `loc`, é possível também escolher mais do que uma coluna. Veja o exemplo a seguir e compare-o com o exemplo anterior:

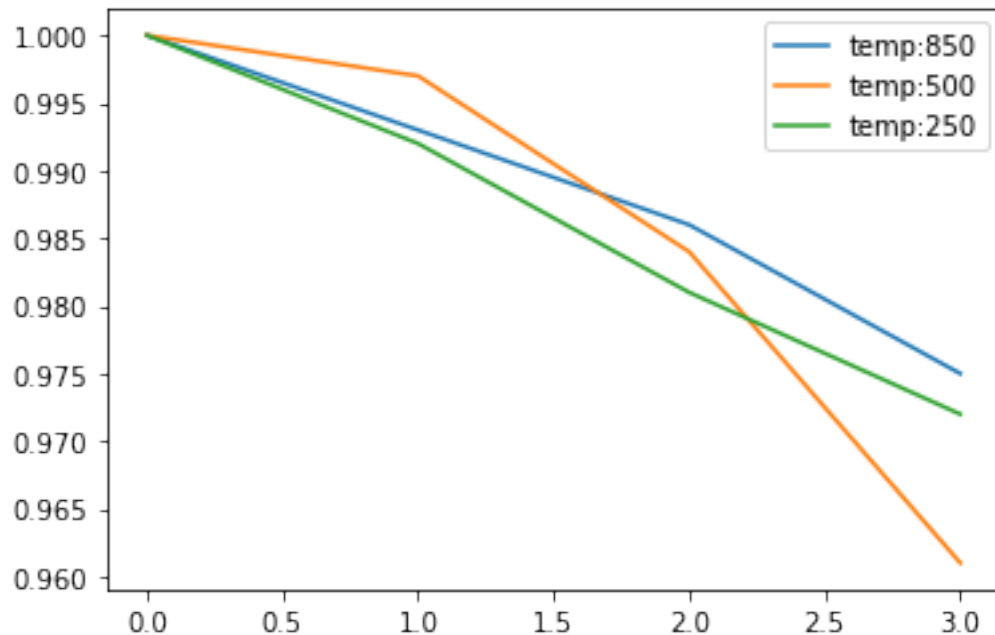
```
[15]: dTable['ACOREXP01_20140805002014080600T.scan'].loc[:,["temp:850", "temp:500",  
↪ "temp:250"]].plot()
```

```
[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f90301ce490>
```



O método `loc` realiza a indexação a partir dos rótulos das colunas. Utilize o método `iloc` para realizar a subseleção a partir dos índices das linhas e colunas:

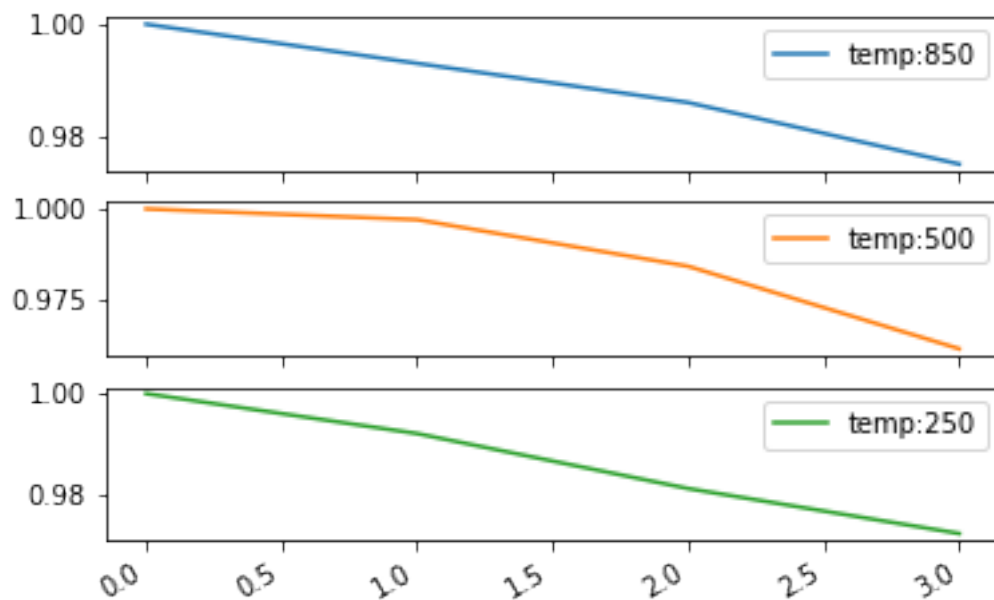
```
[16]: axes = dTable['ACOREXP01_20140805002014080600T.scan'].iloc[:,4:7].plot()
```



Quando múltiplas colunas são selecionadas, pode-se optar pela plotagem em grupo. Para isso, basta passar o argumento `subplots=True` para dentro da função `plot()`:

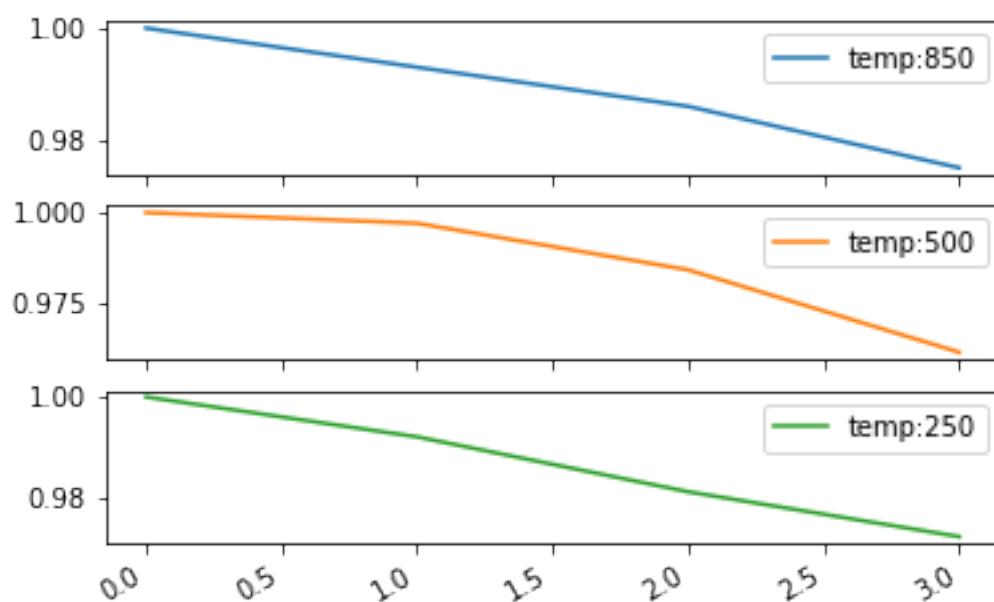
```
[17]: dTable['ACOREXP01_20140805002014080600T.scan'].loc[:,["temp:850", "temp:500",  
↪ "temp:250"]].plot(subplots=True)
```

```
[17]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f90182ec790>,  
            <matplotlib.axes._subplots.AxesSubplot object at 0x7f90109fbcd0>,  
            <matplotlib.axes._subplots.AxesSubplot object at 0x7f9010a07c10>],  
            dtype=object)
```



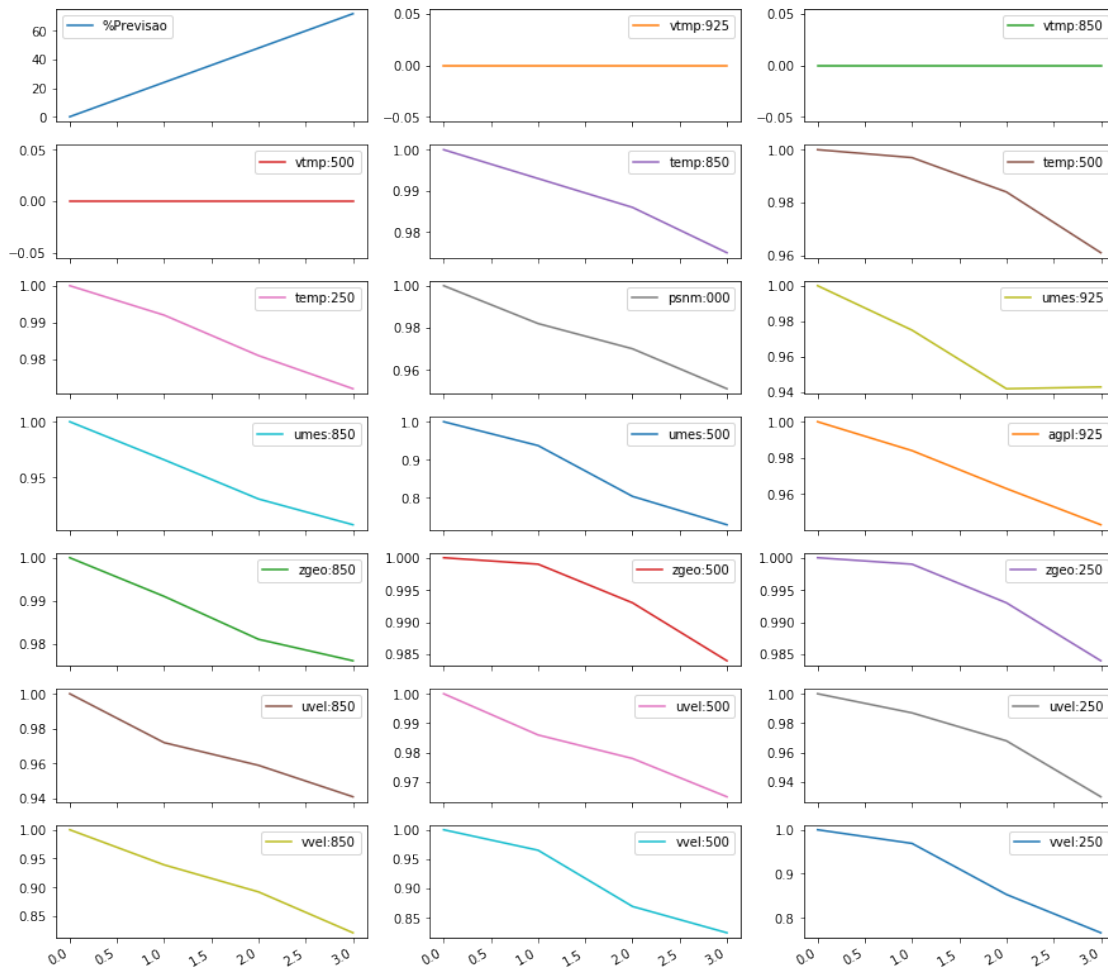
Veja a seguir o mesmo exemplo anterior, mas utilizando o método `iloc`:

```
[18]: axes = dTable['ACOREXP01_20140805002014080600T.scan'].iloc[:,4:7].
      ↪ plot(subplots=True)
```



Outras opções de plotagem com o Matplotlib podem ser passadas também para a função `plot()`, veja a seguir:

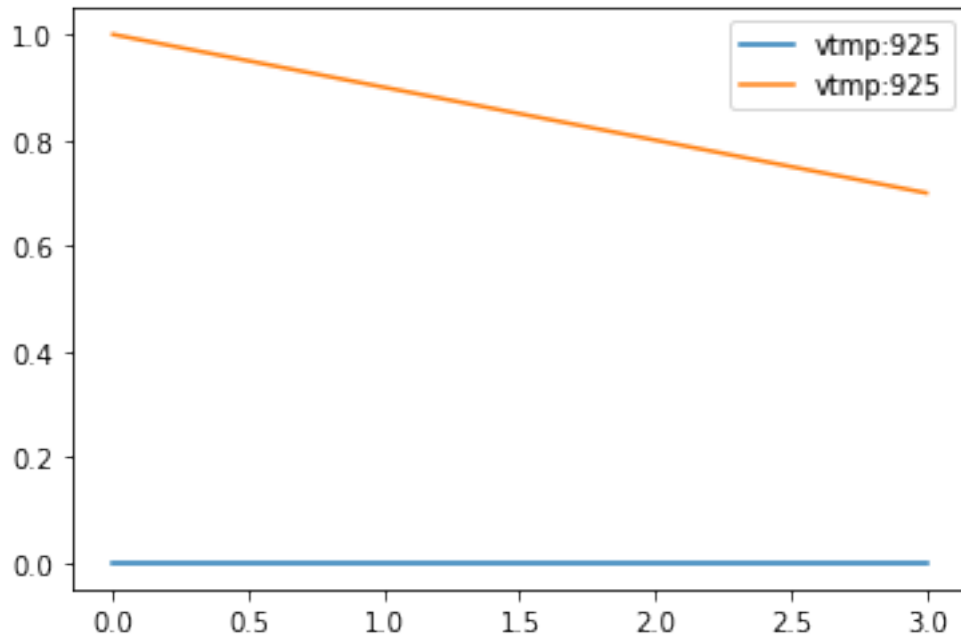
```
[19]: axes = dTable['ACOREXP01_20140805002014080600T.scan'].plot.line(subplots=True,
↳ figsize=(15,15), layout=(7,3), sharex=True)
```



Com mais do que um experimento, é possível também plotá-los no mesmo gráfico. Veja a seguir como plotar duas colunas de dois dataframes diferentes (ie., duas tabelas do SCANTEC), no mesmo gráfico:

```
[20]: df_exp1 = dTable['ACOREXP01_20140805002014080600T.scan'].loc[:,["vtmp:925"]]
df_exp2 = dTable['ACOREXP02_20140805002014080600T.scan'].loc[:,["vtmp:925"]]
ax = df_exp1.plot()
df_exp2.plot(ax=ax)
```

```
[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f90503f0b50>
```



1.3 Funções de plotagem do SCANPLOT

As tabelas do SCANTEC como dataframes do Pandas, permitem o acesso às facilidades associados ao módulo. O SCANPLOT possui também algumas funções de plotagem que permitem a manipulação das tabelas em lotes. A função `plot_lines` realiza a plotagem das tabelas selecionadas a partir da utilização da função `get_dataframe`.

Para chamar a função `plot_lines`, utilize o comando a seguir:

```
[21]: from scanplot import plot_lines
```

Assim como foi feito anteriormente para as outras funções do SCANPLOT, a ajuda da função pode ser acessada com um dos comandos a seguir:

```
[22]: #help(plot_lines)
print(plot_lines.__doc__)
```

```
plot_lines
=====
```

Esta função plota um gráfico de linha a partir de um dicionário de tabelas do SCANTEC.

```
Parâmetros de entrada
-----
```

```
    dTable : objeto dicionário com uma ou mais tabelas do SCANTEC
```

Vars : lista com os nomes e níveis das variáveis
Stats : lista com os nomes das estatísticas a serem processadas
outDir : string com o diretório com as tabelas do SCANTEC

Resultado

Figuras salvas no diretório definido na variável outDir
(SCANTEC/dataout).

Uso

```
from scanplot import plot_lines  
  
plot_lines(dTable,Vars,Stats,outDir)
```

Veja que a função `plot_lines` recebe como parâmetros de entrada o dicionário `dTable`, as listas `Vars` e `Stats` e o diretório de saída `outDir` que será utilizado para salvar as figuras produzidas. Veja a seguir como utilizar a função `plot_lines`:

```
[23]: plot_lines(dTable,Vars,Stats,outDir)
```

