xarray

April 5, 2019

1 Criando um xarray DataSet a partir de dados sintéticos

Este tutorial irá ensinar como criar um Xarray DataSet, que poderá ser utilizado para armazenar informações georeferenciadas. Além da criação do Xarray DataSet, serão mostrados também exemplos simples para a seleção ("slicing") do DataSet e a criação de gráficos espaciais e de linha.

1.1 Importando os módulos principais

No Python os módulos (ou bibliotecas) são pacotes que permitem executar determinadas tarefas. Se você está habituado com o Matlab, entenda os módulos do Python com as "toolboxes" do Matlab.

O "numpy" é provavelmente o módulo mais popular. Ele fornece funções básicas e avançadas para álgebra linear e diversos tipos de cálculos com arrays de n dimensões. Para importá-lo, basta digitar o comando abaixo. Observe que ao longo do script, utilizaremos o mnemônico "np" para chamar as funções relacionadas ao numpy.

```
In [1]: import numpy as np
```

O "pandas" é um módulo bastante popular cuja função principal é facilitar o trabalho com dados tabulados. Além disso, ele também possui características interessantes que permitem a criação de séries numéricas e de dados temporais. O mnemônico utilizado ao longo deste script é o "pd".

```
In [2]: import pandas as pd
```

O "xarray" é um módulo que basicamente expande as funcionalidades do pandas para arrays de n dimensões. A sua principal característica é permitir a georeferenciação dos arrays, e ele é o módulo principal utilizado neste script. O seu mnemônico é o "xr".

```
In [3]: import xarray as xr
```

Para visualizar os arrays criados, iremos utilizar o módulo "matplotlib". Observe que estamos utilizando a função "pyplot" e criando o mnemônico "plt" para a sua utilização.

```
In [4]: from matplotlib import pyplot as plt
```

1.2 Criando as coordenadas temporal e espacial

1.2.1 Latitudes

Para criar um Xarray DataSet, ou seja, um conjunto de dados que tenha referências geográficas, criaremos primeiro uma grade discretizada em pontos de latitude e longitude. Para criar um array com valores de latitude, iremos utilizar a função "linspace" do módulo numpy. Neste exemplo, criaremos 32 pontos de latitude, dentro do intervalo [-90,90].

```
In [5]: lats = np.linspace(-90,90,32)
```

Para visualizar o array criado, você pode utilizar o comando "print()" ou pode simplesmente digitar "lats".

```
In [6]: print("lats:\n",lats)
```

```
lats:
 [-90.         -84.19354839 -78.38709677 -72.58064516 -66.77419355
 -60.96774194 -55.16129032 -49.35483871 -43.5483871  -37.74193548
 -31.93548387 -26.12903226 -20.32258065 -14.51612903  -8.70967742
  -2.90322581   2.90322581   8.70967742  14.51612903  20.32258065
  26.12903226  31.93548387  37.74193548  43.5483871   49.35483871
  55.16129032  60.96774194  66.77419355  72.58064516  78.38709677
  84.19354839  90.        ]
```

1.2.2 Longitudes

Criaremos os pontos de longitude da nossa grade da mesma forma, utilizando a função "linspace" do módulo numpy. Neste exemplo, criaremos um array de 64 pontos de longitude, dentro do intervalo [0,360].

```
In [7]: lons = np.linspace(-180,180,64)
```

Imprima os valores do array "lats" utilizando o comando "print(lons)" ou simplesmente "lons" no prompt abaixo.

```
In [8]: print("lons:\n",lons)
```

```
lons:
 [-180.         -174.28571429 -168.57142857 -162.85714286 -157.14285714
 -151.42857143 -145.71428571 -140.         -134.28571429 -128.57142857
 -122.85714286 -117.14285714 -111.42857143 -105.71428571 -100.
  -94.28571429  -88.57142857  -82.85714286  -77.14285714  -71.42857143
  -65.71428571  -60.          -54.28571429  -48.57142857  -42.85714286
  -37.14285714  -31.42857143  -25.71428571  -20.          -14.28571429
   -8.57142857   -2.85714286    2.85714286    8.57142857   14.28571429
   20.           25.71428571   31.42857143   37.14285714   42.85714286
   48.57142857   54.28571429   60.           65.71428571   71.42857143
   77.14285714   82.85714286   88.57142857   94.28571429  100.
```

```
  105.71428571  111.42857143  117.14285714  122.85714286  128.57142857
  134.28571429  140.          145.71428571  151.42857143  157.14285714
  162.85714286  168.57142857  174.28571429  180.        ]
```

1.2.3 Tempos

Além das coordenadas geográficas de latitude e longitude, criaremos também um array que conterá alguns timesteps que irão caracterizar o Xarray DataSet que estamos criando. Para isso, podemos utilizar a função "date_range" do pandas. Criaremos 4 timesteps a partir da data "2014-11-01", ou seja, a partir de 1 de novembro de 2014.

```
In [9]: times = pd.date_range("2014-11-01",periods=4)
```

Imprime os valores do array "times" criado com o comando "print" no prompt abaixo.

```
In [10]: print("times:\n",times)
```

```
times:
 DatetimeIndex(['2014-11-01', '2014-11-02', '2014-11-03', '2014-11-04'], dtype='datetime64[ns]',
```

1.3 Criando um array de dados sintéticos

Até aqui o que fizemos foi apenas criar arrays que serão utilizados para criar o nosso Xarray DataSet. No entanto, ainda é necessário criar valores que representarão a informação geoespacial que queremos representar. Para isso, podemos utilizar a função "random.randn" do módulo numpy. Esta função gera um array de valores aleatórios e com a dimensão que quisermos. No entanto, para tonar o problema um pouco mais realista, criaremos uma função de duas variáveis que representará uma superfície, de forma que tenhamos valores que seguem a função "seno". Como temos três coordenadas (latitude, longitude e tempo), iremos criar um array com 3 dimensões. Observe que "Z" irá representar a superfície criada e que utilizaremos esta informação para adiconarmos valores aleatórios que serão utilzados para a composição da dimensão temporal. Ao final, somaremos o valor "30" para alcançar valores que sejam representativos da temperatura do ar em graus Celsius.

```
In [11]: def f(x, y):
             return (np.sin(x) + 2*np.sin(y))

         x = lats
         y = lons

         X, Y = np.meshgrid(x, y)
         Z = f(X, Y)
```

```
In [12]: temp = Z + np.random.randn(len(times),len(lons),len(lats)) + 30
```

Imprima os valores do array "temp" e inspecione os valores criados.

```
In [13]: print("temp:\n",temp)
```

```
temp:
 [[[29.5856036  29.59539346 31.65366648 ... 30.65244491 32.8229572
    33.94470663]
   [31.51957679 29.22496836 31.54628848 ... 31.06364704 33.85170519
    33.06664313]
   [31.92466958 30.65809446 32.02730401 ... 32.94866156 33.13020102
    33.17362346]
   ...
   [28.71781981 26.98604006 28.8260933  ... 28.96889362 29.18498539
    29.40758519]
   [27.31874431 27.25555655 28.23894076 ... 29.88168646 30.69569629
    28.86549587]
   [26.86668251 28.20368665 29.46218111 ... 26.77336299 28.60467702
    29.15245825]]

 [[30.50863424 31.0317055  30.3383215  ... 33.82693125 33.5087904
    32.92820935]
   [31.07296091 31.50461295 32.35722897 ... 30.41510378 32.62142429
    33.12253315]
   [31.92232067 32.07728566 31.17521475 ... 31.46527274 34.96786604
    31.2730047 ]
   ...
   [26.68781791 27.80750768 28.53750306 ... 30.7923073  29.38309118
    27.02975504]
   [26.67278287 27.32723944 27.20316797 ... 27.30691012 30.2094
    29.94918774]
   [28.67910217 27.43983576 26.38120476 ... 27.37248177 26.83869482
    30.02664851]]

 [[29.6219706  32.04726416 31.95274233 ... 31.63208511 33.85612171
    34.16859491]
   [30.2615454  31.92151855 31.12171167 ... 31.90151228 31.569101
    32.45054435]
   [30.71155595 31.46463079 33.28218164 ... 31.30335096 32.72643833
    31.15860344]
   ...
   [27.37015026 29.3951669  26.08389579 ... 28.28173002 28.30237753
    29.03092559]
   [26.48446499 28.05702661 28.74950235 ... 26.77319803 28.85457596
    28.92806822]
   [28.93195185 29.17395731 29.06418247 ... 29.99853484 29.71493048
    29.29129307]]

 [[30.67076017 32.02244267 31.72467476 ... 30.28042673 31.91417544
    34.46143828]
   [32.95259026 30.50767438 34.22214717 ... 31.10505962 32.88635296
    32.82868941]
   [31.93641614 32.49066209 33.51717218 ... 31.30129376 33.01785177
```

```
  32.84986118]
 ...
 [27.05949103 28.6934202  27.31651265 ... 28.43192779 29.73591185
  27.75232017]
 [27.53398216 27.62772691 28.26953995 ... 29.29076536 27.56741999
  29.08293774]
 [25.90630454 29.00331387 27.51589408 ... 28.78788228 29.67379732
  29.93954035]]]
```

Para verificar o tamanho do array que criamos, utilize o método "shape":

```
In [14]: temp.shape

Out[14]: (4, 64, 32)
```

Ou seja, temos o array "temp" possui 3 dimensões, sendo a primeira o tempo (4 tempos), a segunda são as longitudes (64 valores) e a última, as latitudes (32 valores).

1.3.1 Criando o Dataset

Com todas as coordenadas criadas e com o array de dados já determinado, podemos inicializar o Xarray Dataset. Para isso, utilizamos a função "DataSet" do xarray:

```
In [15]: ds = xr.Dataset()
```

Digite "ds" para verificar a estrutura básica do Xarray DataSet criado.

```
In [16]: ds

Out[16]: <xarray.Dataset>
         Dimensions:  ()
         Data variables:
             *empty*
```

Observe que o Xarray DataSet não possui nenhum array associado e nenhuma dimensão. Vamos adicionar os arrays criados a este Xarray DataSet. Adicionando a variável "temp" ao Dataset, observe que a ordem das dimensões deve ser a mesma que indexa o array "temp":

```
In [17]: ds["temp"] = (("time","lon","lat"),temp)
```

Verifique novamente a estrutura do Xarray DataSet "ds":

```
In [18]: ds

Out[18]: <xarray.Dataset>
         Dimensions:  (lat: 32, lon: 64, time: 4)
         Dimensions without coordinates: lat, lon, time
         Data variables:
             temp     (time, lon, lat) float64 29.59 29.6 31.65 ... 28.79 29.67 29.94
```

Observe que agora o Xarray DataSet possui as dimensões "lat", "lon" e "time" e que "temp" é reconhecido como uma variável. Apesar disso, as dimensões citadas ainda não possuem coordenadas. Xarrays possuem coordenadas e dimensões. A diferença entre estas duas estruturas é que a primeira representa os valores e a segunda, o tamanho. Logo, quando definimos que o array "lats" possui 32 valores, acabamos por definir que a sua dimensão é 32, e os valores contidos no array representam as coordenadas. Mas observe que as coordenadas devem ser utilizadas para indexar um array que representa uma ou mais variáveis, sendo esta "temp" neste exemplo. Para adicionar as coordenadas espaciais, basta utilizar o método "coords":

```
In [19]: ds.coords["lat"] = ("lat",lats)
```

Verifique novamente a estrutura do Xarray DataSet "ds":

```
In [20]: ds

Out[20]: <xarray.Dataset>
         Dimensions:  (lat: 32, lon: 64, time: 4)
         Coordinates:
           * lat      (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 72.58 78.39 84.19 90.0
         Dimensions without coordinates: lon, time
         Data variables:
             temp     (time, lon, lat) float64 29.59 29.6 31.65 ... 28.79 29.67 29.94
```

Observe que agora o array "lat" aparece como uma coordenada, que representa os 320 valores do array "lats". Vamos adicionar agora as coordenadas do array "lons":

```
In [21]: ds.coords["lon"] = ("lon",lons)

In [22]: ds

Out[22]: <xarray.Dataset>
         Dimensions:  (lat: 32, lon: 64, time: 4)
         Coordinates:
           * lat      (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 72.58 78.39 84.19 90.0
           * lon      (lon) float64 -180.0 -174.3 -168.6 -162.9 ... 168.6 174.3 180.0
         Dimensions without coordinates: time
         Data variables:
             temp     (time, lon, lat) float64 29.59 29.6 31.65 ... 28.79 29.67 29.94
```

Adicionando a coordenada temporal:

```
In [23]: ds.coords["time"] = times

In [24]: ds

Out[24]: <xarray.Dataset>
         Dimensions:  (lat: 32, lon: 64, time: 4)
         Coordinates:
           * lat      (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 72.58 78.39 84.19 90.0
           * lon      (lon) float64 -180.0 -174.3 -168.6 -162.9 ... 168.6 174.3 180.0
           * time     (time) datetime64[ns] 2014-11-01 2014-11-02 2014-11-03 2014-11-04
         Data variables:
             temp     (time, lon, lat) float64 29.59 29.6 31.65 ... 28.79 29.67 29.94
```

Adicionadas as coordenadas de latitude, longitude e tempo, Podemos dar um passo adiante e criar uma nova coordenada que fará alusão à data da condição inicial. Observe que das datas que foram criadas na coordenada "time" (associada à dimensão "times"), estas fazem alusão aos tempos de previsão. Para Adicionando um tempo de referência, utilizamos também o método "coords" e a função "Timestamp" do Pandas para criar uma data:

```
In [25]: ds.coords["analysis_time"] = pd.Timestamp("2014-10-31")

In [26]: ds

Out[26]: <xarray.Dataset>
         Dimensions:        (lat: 32, lon: 64, time: 4)
         Coordinates:
           * lat            (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 78.39 84.19 90.0
           * lon            (lon) float64 -180.0 -174.3 -168.6 ... 168.6 174.3 180.0
           * time           (time) datetime64[ns] 2014-11-01 2014-11-02 ... 2014-11-04
             analysis_time  datetime64[ns] 2014-10-31
         Data variables:
             temp           (time, lon, lat) float64 29.59 29.6 31.65 ... 29.67 29.94
```

Observe que as coordenadas "time" e "reference_time" possuem objetos do tipo "datetime". Este é um detalhe importante, pois os softwares que lêem arquivos NetCDF conseguirão interpretar estas informações como informações temporais.

Para tornar o nosso Xarray DataSet mais completo, podemos incluir metadados que irão descrever os diferentes atributos do DataSet. Para incluir informações de metadados, podemos utilizar o método "attrs" do Xarray:

```
In [27]: ds.attrs["title"] = "Absolute Temperature"

In [28]: ds

Out[28]: <xarray.Dataset>
         Dimensions:        (lat: 32, lon: 64, time: 4)
         Coordinates:
           * lat            (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 78.39 84.19 90.0
           * lon            (lon) float64 -180.0 -174.3 -168.6 ... 168.6 174.3 180.0
           * time           (time) datetime64[ns] 2014-11-01 2014-11-02 ... 2014-11-04
             analysis_time  datetime64[ns] 2014-10-31
         Data variables:
             temp           (time, lon, lat) float64 29.59 29.6 31.65 ... 29.67 29.94
         Attributes:
             title:    Absolute Temperature
```

Observe que uma nova seção "Attributes" foi adicionada à estrutura do Xarray DataSet criado, com um rótulo "title" e uma string associada "Absolute Temperature". Agora, como já adicionamos o nome da variável que buscamos representar, vamos adicionar um novo metadado, que irá representar a unidade em que os dados estão sendo representados:

```
In [29]: ds.attrs["unit"] = "Degrees Celsius"
```

```
In [30]: ds

Out[30]: <xarray.Dataset>
         Dimensions:        (lat: 32, lon: 64, time: 4)
         Coordinates:
           * lat            (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 78.39 84.19 90.0
           * lon            (lon) float64 -180.0 -174.3 -168.6 ... 168.6 174.3 180.0
           * time           (time) datetime64[ns] 2014-11-01 2014-11-02 ... 2014-11-04
             analysis_time  datetime64[ns] 2014-10-31
         Data variables:
             temp           (time, lon, lat) float64 29.59 29.6 31.65 ... 29.67 29.94
         Attributes:
             title:    Absolute Temperature
             unit:     Degrees Celsius
```

Observe que os novos metadados são acrescentados à lista de atributos do Xarray DataSet. Um outro metadado que podemos acrescentar, é a resolução da grade que estamos representando. Considerando que a grade que estamos representado é uma grade regular retangular, podemos utilizar a dimensão das longitudes (ou latitudes) para calcular a resolução da grade:

```
In [31]: ds.attrs["resolution"] = 360 / (len(lons)-1)

In [32]: ds

Out[32]: <xarray.Dataset>
         Dimensions:        (lat: 32, lon: 64, time: 4)
         Coordinates:
           * lat            (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 78.39 84.19 90.0
           * lon            (lon) float64 -180.0 -174.3 -168.6 ... 168.6 174.3 180.0
           * time           (time) datetime64[ns] 2014-11-01 2014-11-02 ... 2014-11-04
             analysis_time  datetime64[ns] 2014-10-31
         Data variables:
             temp           (time, lon, lat) float64 29.59 29.6 31.65 ... 29.67 29.94
         Attributes:
             title:       Absolute Temperature
             unit:        Degrees Celsius
             resolution:  5.714285714285714
```

Com isso, finalizamos a construção do Xarray DataSet. Observe que incluímos as dimensões, coordenadas, array e atributos ao DataSet. Na seção a seguir, iremos adicionar uma nova variável.

1.3.2 Acrescentando uma nova variável (com as mesmas dimensões)

Suponha agora que queiramos acrescentar uma nova variável, que possua as mesas dimensões que a variável "temp". A nova variável, denominada "psnm", possuirá valores semelhantes à pressão em superfície. Para obtermos os valores, faremos um procedimento semelhante ao da variável "temp", sendo que ao final somaremos "1000" para que os valores sejam semelhantes à pressão em superfície:

```
In [33]: psnm = Z + np.random.randn(len(times),len(lons),len(lats)) + 1000
```

```
In [34]: print("psnm:\n",psnm)
```

```
psnm:
 [[[ 999.40415969 1001.55853215  999.54094001 ... 1002.37721556
    1001.22036326 1002.99650963]
  [1000.47975097 1001.71253382 1001.391195   ... 1001.66193109
    1001.71558359 1003.26080808]
  [1001.7748042   998.4742789  1001.3146098  ... 1001.88144685
    1001.390753   1003.99294537]
  ...
  [ 997.05663093  997.52608672  995.72785444 ...  998.34906265
    998.46544478  999.69653764]
  [ 996.49891902  996.16477127  997.30387139 ...  996.02322298
    999.96929443  997.99633115]
  [ 996.6789693   999.42277933  998.66404181 ...  998.02568199
    998.3971592  1000.10020458]]

 [[1000.38429178  999.80177721 1000.85027524 ... 1002.19333971
    1003.00770923  999.85744485]
  [1000.9374705  1000.85856606 1001.55669165 ... 1000.69807546
    1002.23194156 1003.04591002]
  [1002.00576654 1000.26656937 1002.45924879 ... 1002.65505977
    1002.42448058 1002.81604997]
  ...
  [ 998.49340753  999.03806475  997.40765215 ...  996.88756099
    997.00049028  999.06539219]
  [ 996.66824646  997.65506447  998.11387116 ...  998.29058673
    999.71479481  998.91002293]
  [ 997.1997793   998.4362567   998.5250194  ...  999.29221493
    997.93856213  999.20536365]]

 [[1000.70411575 1000.25117102 1001.90000917 ... 1003.69525909
    1001.96115445 1000.71588654]
  [1002.70394431 1001.33118153 1002.54627654 ... 1003.86568638
    1002.1460744  1001.00068232]
  [ 999.77177862 1002.77899347 1003.15148338 ... 1004.97106714
    1002.65599087 1004.20252828]
  ...
  [ 997.20294244  998.30540927  998.80909199 ...  998.01447425
    998.48068056 1000.68550431]
  [ 998.07689564  996.77628116  998.98313627 ...  999.39764724
    999.98420079  998.23124549]
  [ 998.21464069  998.24797319  996.26039532 ... 1000.68320578
    999.66346281  997.8304903 ]]

 [[1001.88679779 1001.69084457 1002.56800721 ... 1002.85829187
    1000.91274196 1002.44169472]
  [1001.86305811 1002.04165895 1001.05217502 ... 1003.24910272
```

```
  1005.17068118 1001.02093342]
 [1000.13649621  999.43359782 1001.60410865 ... 1000.82224236
  1002.0284485  1003.53631642]
 ...
 [ 996.35138223  999.15106797  999.05950638 ...  998.6177206
   997.95400871  998.63935665]
 [ 995.1027061   998.16372433  997.01193202 ...  998.4064564
   998.65657461  997.69803096]
 [ 996.82802713  997.17127918  999.39368752 ...  998.41529052
   998.23326682 1000.77514576]]]
```

Para adicionar a variável "psnm" ao Xarray DataSet "ds" criado, basta indicar "psnm" como sendo uma tupla que conterá as mesmas dimensões e coordenadas. Acrescentando a variável "psnm" ao Dataset "ds":

```
In [35]: ds["psnm"] = (("time","lon","lat"),psnm)

In [36]: ds

Out[36]: <xarray.Dataset>
         Dimensions:        (lat: 32, lon: 64, time: 4)
         Coordinates:
           * lat            (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 78.39 84.19 90.0
           * lon            (lon) float64 -180.0 -174.3 -168.6 ... 168.6 174.3 180.0
           * time           (time) datetime64[ns] 2014-11-01 2014-11-02 ... 2014-11-04
             analysis_time  datetime64[ns] 2014-10-31
         Data variables:
             temp           (time, lon, lat) float64 29.59 29.6 31.65 ... 29.67 29.94
             psnm           (time, lon, lat) float64 999.4 1.002e+03 ... 998.2 1.001e+03
         Attributes:
             title:       Absolute Temperature
             unit:        Degrees Celsius
             resolution:  5.714285714285714
```

1.4 Plotando a grade criada

Agora que já temos um Xarray DataSet, podemos selecionar a variável que quisermos, para uma determinado tempo de previsão, ou ainda para uma região dentro do domínio e plotar o resultado.

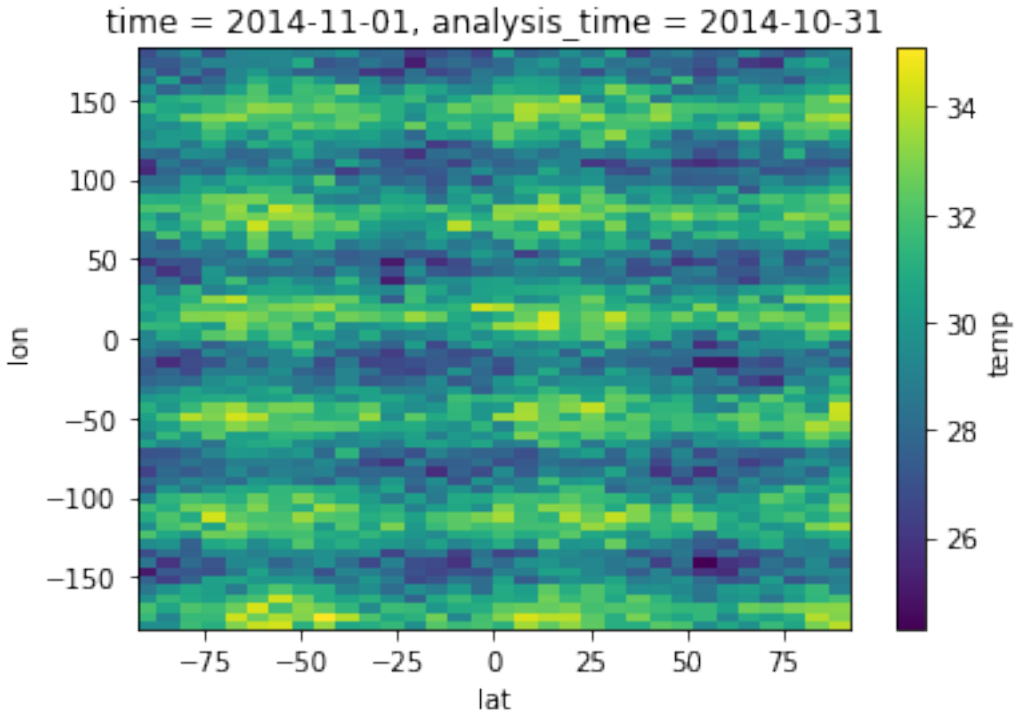
Primeiro, vamos selecionar a variável "temp", que representa a temperatura do ar, e vamos também selecionar o primeiro tempo de previsão. Para isso, indicamos a variável "temp" e utilizamos o método "isel" do Xarray, indicando o tempo. O método "isel" indica que podemos fazer seleções utilizando números inteiros, ou seja, quando fazemos "isel(time=0)", estamos selecionando o primeiro valor da coordenada "time":

```
In [37]: temp_t0=ds["temp"].isel(time=0)

In [38]: temp_t0
```

```
Out[38]: <xarray.DataArray 'temp' (lon: 64, lat: 32)>
         array([[29.585604, 29.595393, 31.653666, ..., 30.652445, 32.822957, 33.944707],
                [31.519577, 29.224968, 31.546288, ..., 31.063647, 33.851705, 33.066643],
                [31.92467 , 30.658094, 32.027304, ..., 32.948662, 33.130201, 33.173623],
                ...,
                [28.71782 , 26.98604 , 28.826093, ..., 28.968894, 29.184985, 29.407585],
                [27.318744, 27.255557, 28.238941, ..., 29.881686, 30.695696, 28.865496],
                [26.866683, 28.203687, 29.462181, ..., 26.773363, 28.604677, 29.152458]])
         Coordinates:
           * lat            (lat) float64 -90.0 -84.19 -78.39 -72.58 ... 78.39 84.19 90.0
           * lon            (lon) float64 -180.0 -174.3 -168.6 ... 168.6 174.3 180.0
             time           datetime64[ns] 2014-11-01
             analysis_time  datetime64[ns] 2014-10-31
```

Para plotar, basta utilizar o método "plot()":

```
In [39]: temp_t0.plot()

Out[39]: <matplotlib.collections.QuadMesh at 0xa9c0a5ac>
```



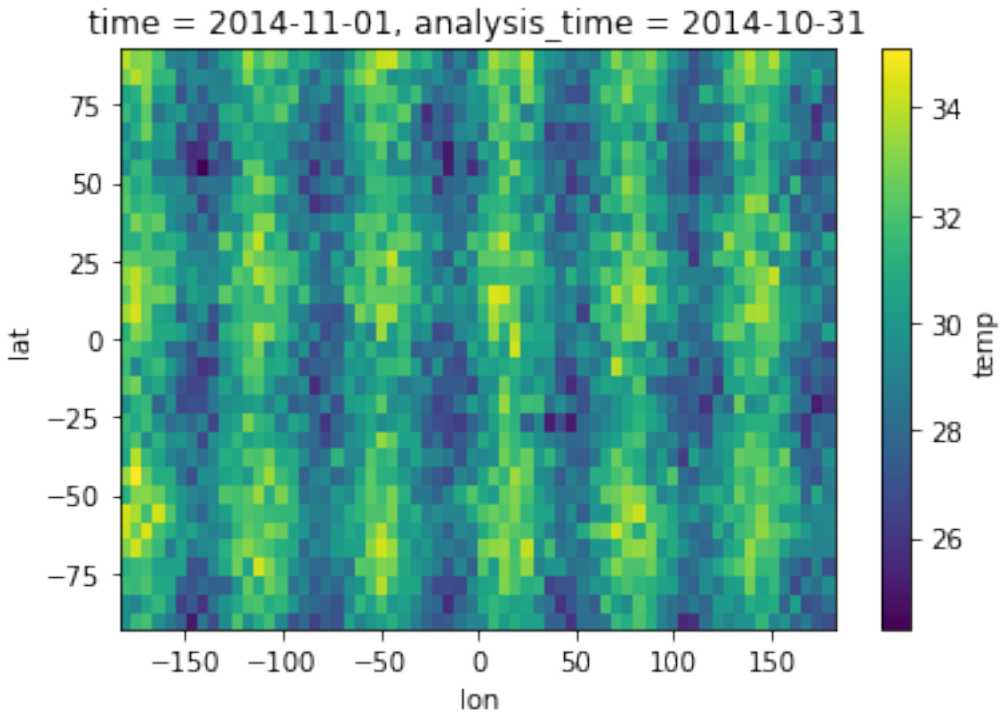
Na figura acima, observe que as coordenadas "lat" e "lon" estão trocadas. Queremos um gráfico que represente as latitudes no eixo y e as longitudes no eixo x. Para fazer isso, utilizamos o método "transpose" do Xarray:

```
In [40]: temp_t0_transp = temp_t0.transpose("lat","lon")
```

Para plotar, basta utilizar o método "plot()":

```
In [41]: temp_t0_transp.plot()
```

```
Out[41]: <matplotlib.collections.QuadMesh at 0xa9b23b0c>
```



Observe na figura que os metadatas que adicionamos como atributos foram todos utiizados automaticamente como nomes dos eixos, título e legenda.

1.5 Plotando gráficos de linhas

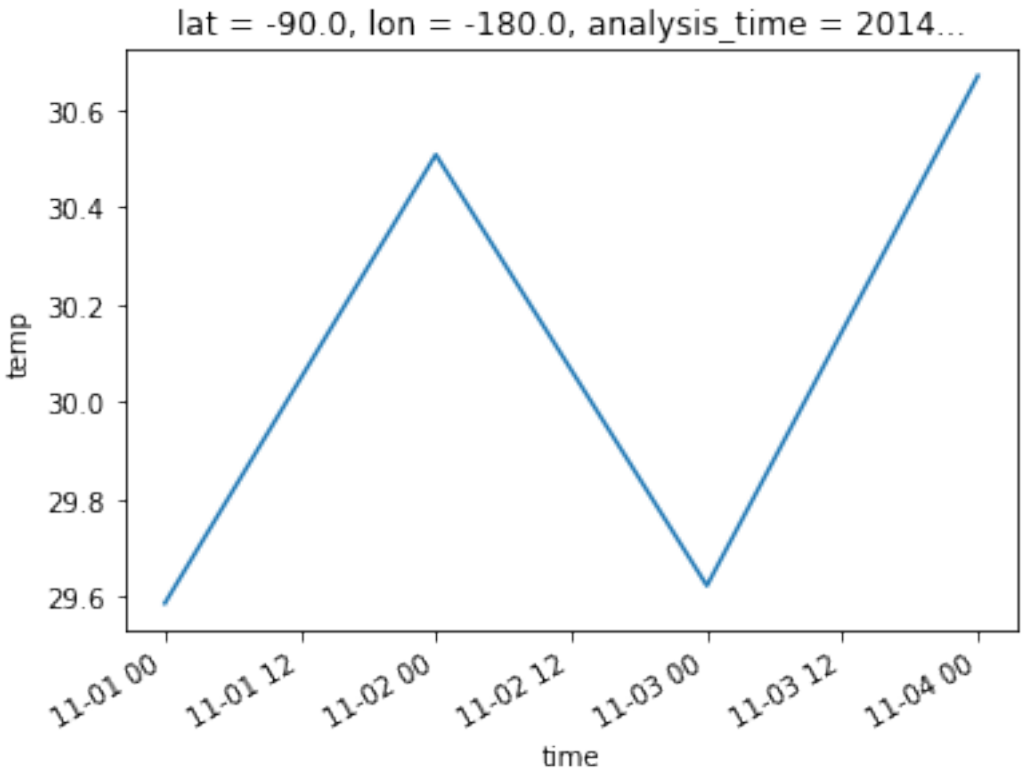
Utilizando o método "isel" do Xarray, é possível selecionar pontos de latitude e longitude e plotar uma série temporal. Veja o exemplo a seguir:

```
In [42]: temp_latlon = ds["temp"].isel(lat=0,lon=0)
```

```
In [43]: temp_latlon
```

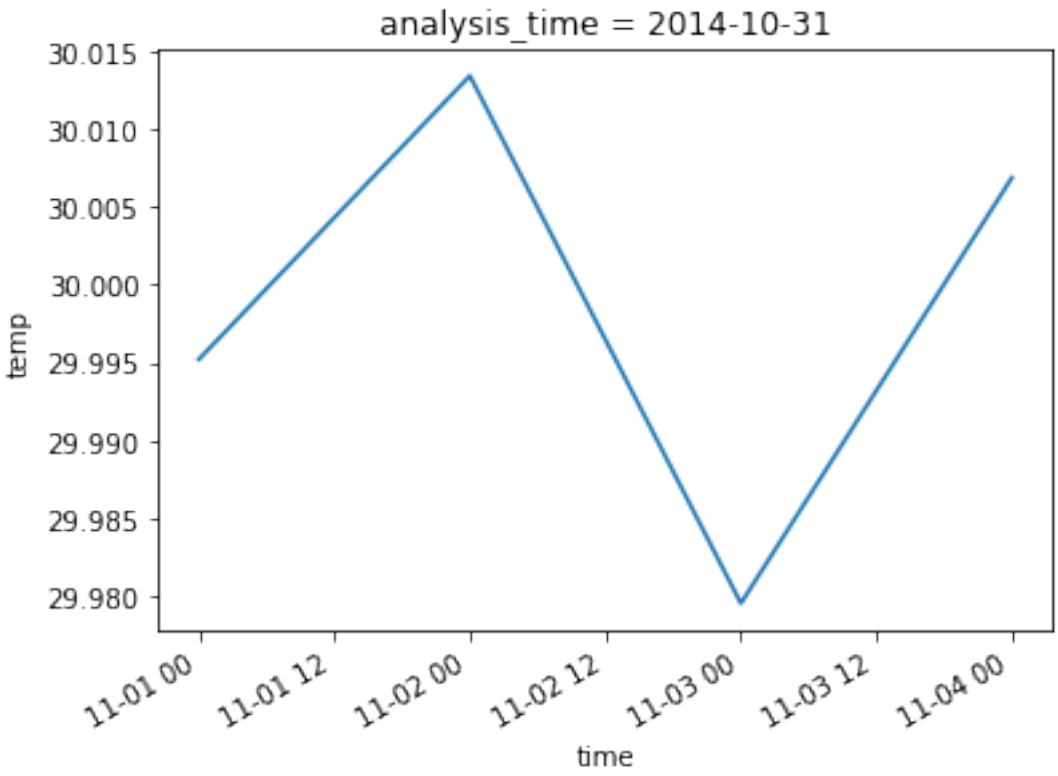
```
Out[43]: <xarray.DataArray 'temp' (time: 4)>
         array([29.585604, 30.508634, 29.621971, 30.67076 ])
         Coordinates:
             lat            float64 -90.0
             lon            float64 -180.0
           * time           (time) datetime64[ns] 2014-11-01 2014-11-02 ... 2014-11-04
             analysis_time  datetime64[ns] 2014-10-31
```

```
In [44]: temp_latlon.plot()

Out[44]: [<matplotlib.lines.Line2D at 0xa9a5512c>]
```



Além do método "isel", é possível utiizar também o método "mean", para médias espaciais. Por exemplo, calculando-se médias ao longo das coordenadas "lat" e "lon":

```
In [45]: temp_latlon_mean = ds["temp"].mean(dim=["lat","lon"])

In [46]: temp_latlon_mean

Out[46]: <xarray.DataArray 'temp' (time: 4)>
         array([29.995192, 30.013417, 29.979518, 30.006876])
         Coordinates:
           * time           (time) datetime64[ns] 2014-11-01 2014-11-02 ... 2014-11-04
             analysis_time  datetime64[ns] 2014-10-31

In [47]: temp_latlon_mean.plot()

Out[47]: [<matplotlib.lines.Line2D at 0xa9a1dd6c>]
```



1.6 Salvando o Xarray DataSet em um arquivo NetCDF

Uma vantagem que o Xarray possui é salvar o DataSet em um arquivo NetCDF. Para fazer isso, basta utilizar a função "to_netcdf" do Xarray:

```
In [48]: ds.to_netcdf("ds.nc")
```

Podemos também salvar em um arquivo NetCDF apenas a variável "test", que contém um subset do Xarray DataSet "ds". Para fazer isso, procedemos da mesma forma:

```
In [49]: temp_latlon.to_netcdf("temp.nc")
```

Caso você queira abrir o arquivo NetCDF gravado em disco, você pode utilizar a função "open_dataset" do Xarray:

```
In [50]: ds_temp = xr.open_dataset("temp.nc")
```

Dessa forma criamos o Xarray DataSet "ds_test" que contém a informação do arquivo "test.nc". Compare a estutura deste Xarray DataSet com a estrutura do Xarray DataSet "test":

```
In [51]: ds_temp
```

```
Out[51]: <xarray.Dataset>
         Dimensions:        (time: 4)
         Coordinates:
           * time           (time) datetime64[ns] 2014-11-01 2014-11-02 ... 2014-11-04
             lat            float64 ...
             lon            float64 ...
             analysis_time  datetime64[ns] ...
         Data variables:
             temp           (time) float64 ...
```