Análise topográfica em Python

Luis Moura

2019-08-02

Criação de **Perfis Topográficos** e **Curvas de Nível**, através do uso de Python.

1 Resumo

Criação de **Perfis Topográficos** e **Curvas de Nível**, através do uso de Python.

Dois exemplos são apresentados:

- Exemplo simples
- E um exemplo mais complexo

Ambos os exemplos, apresentam uma forma rápida e simples, de desenhar perfis em 3D e curvas de nível.

O código utilizado, foi adaptado do website Density and Contour Plots | Python Data Science Handbook e de um autor desconhecido.

2 Perfis Topográficos e Curvas de Nível

Existem vários softwares no mercado especializados em levantamentos topográficos e de modo nenhum, Python é um substituto para eles, no entanto, existem situações em que o uso de Python é justificado. No meu caso particular, não tenho nenhum software topográfico instalado nos meus computadores, mas se precisar de fazer um levantamento (simples), é bom saber que tenho uma ferramenta que o pode fazer. E o melhor de tudo, é grátis, sem ter que gastar centenas ou milhares, em licenças anuais.

2.1 Exemplo simples

Um exemplo básico que serve de introdução ao código. Para a criação de modelo topográfico, são necessárias 3 coordenadas para cada ponto. Neste caso, serão usados 15 pontos, com as coordenadas x, y e z.

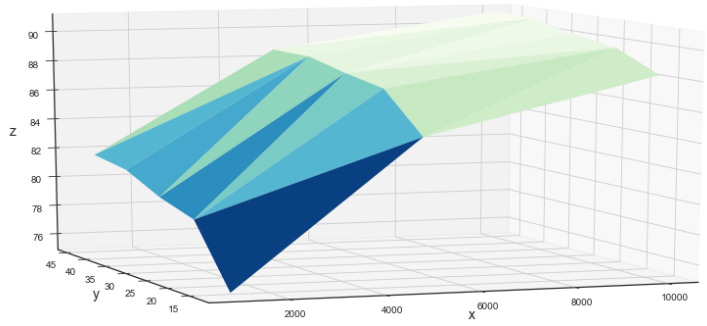
```
x = [1000,1000,1000,1000,1000,5000,5000,5000,5000,5000,
     10000,10000,10000,10000,10000]
```

```
y = [13,21,29,37,45,13,21,29,37,45,13,21,29,37,45]
z = [75.2,79.21,80.02,81.2,81.62,84.79,87.38,87.9,88.54,
     88.56,88.34,89.66,90.11,90.79,90.87]
```

E a criação de um modelo 3D, é feito a partir das coordenadas previamente definidas:

```
fig = plt.figure(figsize=(13,6.5))
ax = fig.gca(projection='3d')
ax.plot_trisurf(x, y, z, cmap=cm.GnBu_r, linewidth=0.2)
ax.view_init(10, -110)
#label axes
ax.set_xlabel('x', fontsize=14)
ax.set_ylabel('y',fontsize=14)
ax.set_zlabel('z',fontsize=14)
plt.tight_layout()
plt.show()
```

Figura 1: Modelo simples em 3D



Mas em vez de um perfil 3D, pode optar-se por um 2D. Usando os mesmos 15 pontos utilizadas na criação da imagem anterior:

```
def plot_contour(x,y,z,resolution = 50,contour_method='linear'):
    resolution = str(resolution)+'j'
    X,Y = np.mgrid[min(x):max(x):complex(resolution),
↪   min(y):max(y):complex(resolution)]
    points = [[a,b] for a,b in zip(x,y)]
    Z = griddata(points, z, (X, Y), method=contour_method)
    return X,Y,Z

X,Y,Z = plot_contour(x,y,z,resolution =
↪   50,contour_method='linear')

with plt.style.context("seaborn-white"):
```

```
    fig, ax = plt.subplots(figsize=(13,8))
    ax.scatter(x,y, color="black", linewidth=9,
↪   edgecolor="ivory", s=50)
    contours=ax.contourf(X,Y,Z,20, cmap='GnBu_r',origin="lower")
    plt.colorbar(contours, shrink=0.67,label="Altitude [m]")
    plt.clabel(contours, inline=True, fontsize=8, fmt='%.1f')
```

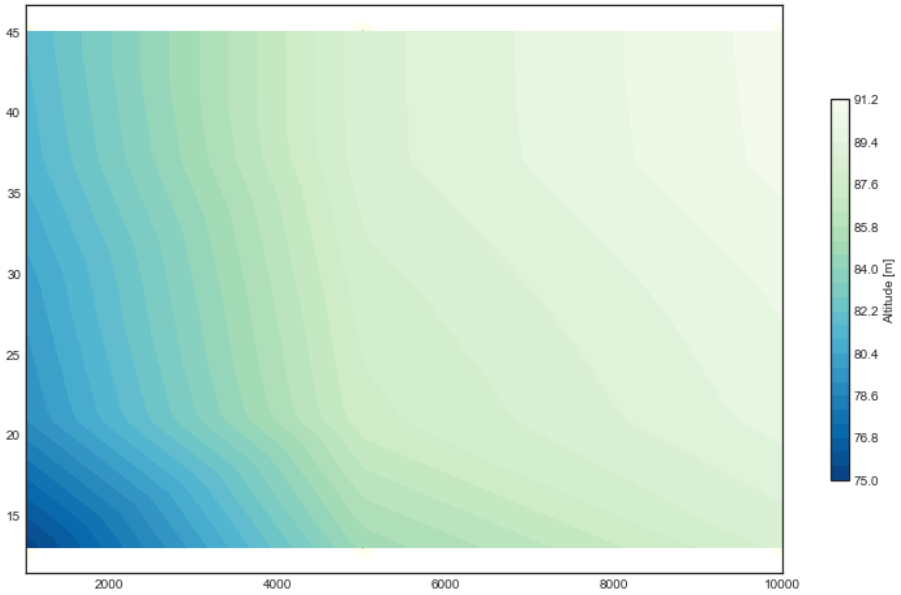


Figura 2: Modelação simples

Assim, e de uma forma simples, é criado um modelo 2D, que através da utilização de cores, torna possível a identificação do comportamento topográfico.

Este será o tipo de modelo mais simples. No entanto, existe sempre a possibilidade da criação de modelos mais complexos[1].

[1] É possível a criação em `Python` de código que permite a análise e criação de modelos complexos. No entanto, existe um limite (em tempo e custo) para quando deixa de ser vantajoso o desenvolvimento desse mesmo código em detrimento de software já existente.

2.2 Exemplo mais complexo

A base de dados encontra-se guardada em Dropbox, e é composta por 85 pontos.

Os primeiros 5 pontos, são apresentados na tabela seguinte.

x	y	z
12100	8300	37.1531
5300	8700	31.4993
3500	13900	36.9185
5100	1900	24.0156
9900	13700	35.0411

A leitura da base de dados é feita através de Pandas: Python Data Analysis Library

```
df =
↪  pd.read_csv('https://www.dropbox.com/s/6dyfc4fl5slhgry/ZoneA.dat?raw=1',
                sep=' ',
                header=9,
                usecols=[0, 1, 2],
                names=['x', 'y', 'thick']
               )
df.rename(columns={'thick':'z'},
                inplace=True)
```

Nota: Código completo no final deste post.

Uma vez obtida a informação da base de dados, é possível a representação das curvas de nível.

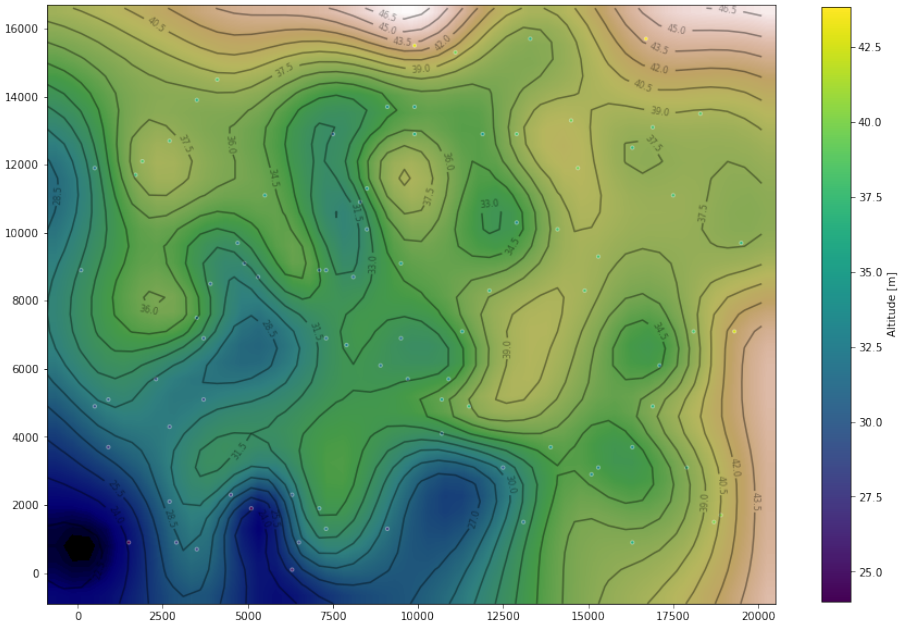


Figura 3: Curvas de nível

E usando a mesma base de dados, pode ser criado um modelo 3D.

```
fig = plt.figure(figsize=(15,7.5))
ax = fig.gca(projection='3d')
triang = mtri.Triangulation(df.x, df.y)
ax.plot_trisurf( triang ,df.z, cmap="gist_earth",
↪  linewidth=0.2,antialiased=True, shade=False)
ax.view_init(20, -75)
plt.title('Elevação 3D', fontsize=16)
#label axes
ax.set_xlabel('x', fontsize=14)
ax.set_ylabel('y',fontsize=14)
ax.set_zlabel('z',fontsize=14)
plt.tight_layout()
plt.show()
```

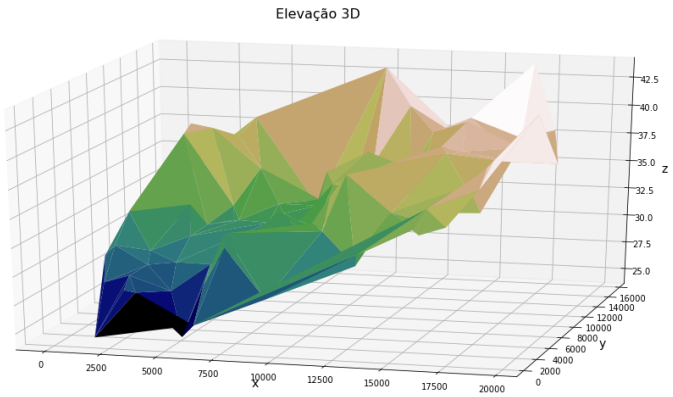


Figura 4: Modelo em 3D

Assim, de uma forma simples e com uma dúzia de linhas de código, é possível a criação de modelos que podem ser facilmente incorporados em relatórios. Qualquer um dos modelos pode ser guardado como PDF, e para isso basta adicionar ao final do código: `plt.savefig("nome.pdf")`

Tudo nestas imagens pode ser alterado, desde orientação da imagem, cores, qualidade dos detalhes, texto, etc. Logicamente, uma maior complexidade, requer um maior número de linhas de código, mas se o principal objectivo é a criação de bons modelos da forma o mais simples possível, os dois exemplos em cima são um bom ponto de partida.

3 Código

Código completo usado neste post

```
# -*- coding: utf-8 -*-

# Livrarias:
# %%
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import cm
plt.style.use('seaborn-white')
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline


#####################################################
# Exemplo 1
#####################################################
# %%
x = [1000,1000,1000,1000,1000,5000,5000,5000,5000,5000,
```

```
     10000,10000,10000,10000,10000]
y = [13,21,29,37,45,13,21,29,37,45,13,21,29,37,45]
z = [75.2,79.21,80.02,81.2,81.62,84.79,87.38,87.9,88.54,
     88.56,88.34,89.66,90.11,90.79,90.87]

fig = plt.figure(figsize=(13,6.5))
ax = fig.gca(projection='3d')
ax.plot_trisurf(x, y, z, cmap=cm.GnBu_r, linewidth=0.2)
ax.view_init(10, -110)
#label axes
ax.set_xlabel('x', fontsize=14)
ax.set_ylabel('y',fontsize=14)
ax.set_zlabel('z',fontsize=14)
plt.tight_layout()
plt.show()


#####################################################
# Exemplo 1 mapa topográfico
#####################################################

# %%
import pandas as pd
from scipy.interpolate import griddata

def plot_contour(x,y,z,resolution = 50,contour_method='linear'):
    resolution = str(resolution)+'j'
    X,Y = np.mgrid[min(x):max(x):complex(resolution),
↪   min(y):max(y):complex(resolution)]
    points = [[a,b] for a,b in zip(x,y)]
    Z = griddata(points, z, (X, Y), method=contour_method)
    return X,Y,Z

X,Y,Z = plot_contour(x,y,z,resolution =
↪   50,contour_method='linear')


with plt.style.context("seaborn-white"):
    fig, ax = plt.subplots(figsize=(13,8))
    ax.scatter(x,y, color="black", linewidth=9,
↪   edgecolor="ivory", s=50)
    contours=ax.contourf(X,Y,Z,20, cmap='GnBu_r',origin="lower")
    plt.colorbar(contours, shrink=0.67,label="Altitude [m]")
    plt.clabel(contours, inline=True, fontsize=8, fmt='%.1f')

#####################################################
# Exemplo 2
#####################################################
```

```
# %%
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import Rbf
%matplotlib inline

# Load the data.
df =
↪  pd.read_csv('https://www.dropbox.com/s/6dyfc4fl5slhgry/ZoneA.dat?raw=1',
                sep=' ',
                header=9,
                usecols=[0, 1, 2],
                names=['x', 'y', 'thick']
               )

df.rename(columns={'thick':'z'},
                inplace=True)

pandas_df_to_markdown_table(df.head())


######################################################
# %%
# Build a regular grid with 500-metre cells.
extent = x_min, x_max, y_min, y_max = [df.x.min()-1000,
↪  df.x.max()+1000,
                                       df.y.min()-1000,
↪  df.y.max()+1000]
grid_x, grid_y = np.mgrid[x_min:x_max:500, y_min:y_max:500]

# Make the interpolator and do the interpolation.
rbfi = Rbf(df.x, df.y, df.z)
di = rbfi(grid_x, grid_y)

# Make the plot.
plt.figure(figsize=(15,15))
plt.imshow(di.T, origin="lower",
↪  extent=extent,cmap='gist_earth',aspect=1,interpolation =
↪  'bilinear')
cb = plt.scatter(df.x, df.y, s=10,
                c=df.z,
                edgecolor='#ffffff66')
plt.colorbar(cb, shrink=0.67,label="Altitude [m]")

params = dict(linestyles='solid', colors=['black'], alpha=0.4)
contours = plt.contour(grid_x, grid_y, di, 20, **params)
plt.clabel(contours, inline=True, fontsize=8, fmt='%.1f')
```

```
plt.show()


#####################################################
# %%
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import matplotlib.tri as mtri

fig = plt.figure(figsize=(15,7.5))
ax = fig.gca(projection='3d')
triang = mtri.Triangulation(df.x, df.y)
ax.plot_trisurf( triang ,df.z, cmap="gist_earth",
↪  linewidth=0.2,antialiased=True, shade=False)
ax.view_init(20, -75)
plt.title('Elevação 3D', fontsize=16)
#label axes
ax.set_xlabel('x', fontsize=14)
ax.set_ylabel('y',fontsize=14)
ax.set_zlabel('z',fontsize=14)
plt.tight_layout()
plt.show()
```