ENSG Géomatique

Utilisation de données vecteur avec Python

Clément Delgrange
03/2018

Table des matières

1 Objectifs

— Connaître les différentes primitives géométriques
— Manipuler des objets géolocalisés et effectuer quelques opérations spatiales
— Visualiser les données géolocalisées, créer des cartes statiques et intéractives
— Ouvrir des fichiers géographiques de différents formats, dans différentes projections
— Effectuer des requêtes spatiales

2 Introduction

La première partie du cours nous a permis d'établir un panorama des possibilités offertes par l'ensemble des techniques de programmation SIG. Dans ce chapitre nous nous proposons d'aborder la programmation SIG en laissant le SIG bureautique traditionnel de côté. Au lieu de cela, nous préférons nous concentrer sur la base de notre métier : la donnée géographique. Nous commencerons ainsi par un bref rappel des modes de preprésentation de l'information géographique. Puis nous verrons comment le langage de programmation Python nous permet de manipuler cette information géographique : manipulation de géométries vecteur, opérations spatiales, gestion des projections, représentation, etc. Nous nous situerons donc ici plutôt dans la dernière des catégories de développements SIG que nous avons établit dans la première partie de ce cours, les applications autonomes, même si les techniques apprises ici pourrons nous être utiles pour développer, par exemple, des plugins pour un SIG.

2.1 Le choix de Python comme langage de programmation

Dans cette partie, comme dans beaucoup de suivantes, nous utiliserons exclusivement le langage de programmation Python. Aujourd'hui Python est probablement le langage de programmation le plus utilisé en géomatique. L'écosystème Python s'est enrichi ses dernières années de nombreuses libraires à visées géospatiales. C'est aussi le langage que les SIG ont choisi depuis plusieurs années pour exécuter des scripts de géotraitements (ArcGIS, QGIS, etc.). Nous pouvons évoquer plusieurs raisons à cela :

— sa facilité d'utilisation et d'apprentissage pour des "non informaticiens" ;
— il est gratuit et open-source ;
— il est assez facilement intégrable à d'autres langages (`C`, `C++`) ;
— de nombreuses libraires de calculs numériques existent (`numpy`, `scipy`, `pandas`, etc.) ;
— les libraires fondamentales de la géomatique (`PROJ.4`, `GDAL`, etc.) bénéficient de bindings Python.

3 Représentation de l'information géographique

3.1 Raster vs. vecteur

Deux façons de stocker les données géolocalisées : le mode vecteur et le mode raster. Ces deux modes permettent de représenter la même information mais le font de manière différente.

En mode raster, le monde réel est représenté sous la forme de matrice où chaque case prend une valeur reflétant la valeur de l'information à représenter. Des matrices multi-dimentionnelles (ie. à plusieurs bandes) sont fréquement utilisées pour représenter les phénomènes complèxes.

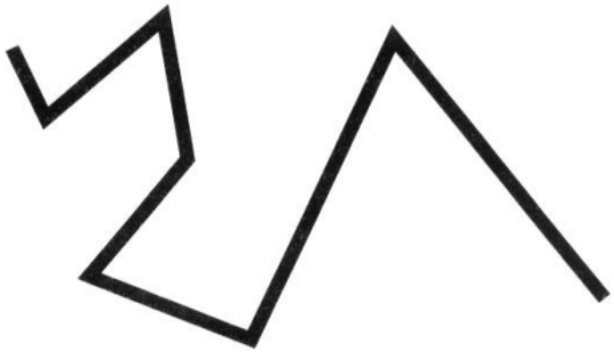

FIGURE 1 – Représentation vecteur

Le mode vecteur représente l'information géographique sous forme de géométries.

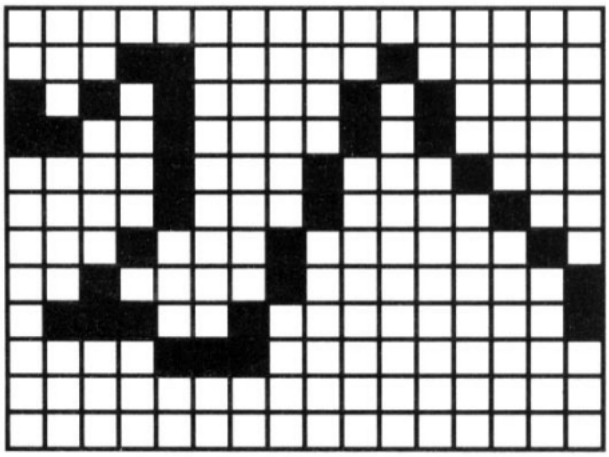

FIGURE 2 – Représentation raster

Les géométries peuvent être dotées d'un sémantique (ie. des attributs) et liées entre elles grace à des liens de topologies (ex : deux routes données sont connectées en un carrefour, ne sont pas connectées, se croisent à un pont mais ne sont pas connectées, etc.).

Si les rasters sont plus faciles à créer, l'analyse des données vecteur est plus aisée. Dans la suite de cette partie nous nous focaliserons sur les données vecteur, tandis que la prochaine partie reviendra sur les données raster.

3.2 Les primitives géométriques

L'information géographique vecteur * Point * MultiPoint * LineString * LineaRing * MultiLineString * Polygon * MultiPolygon * Collection

Validité des géométries :

— polygones croisées
— spikes
— trou dont le contour touche le contour extérieur du polygon en plus d'un points
— etc. [1]

Parler de la lib GEOS / parenthèse sur l'OSGEO

3.3 Les opérations spatiales

— distance
— buffer
— rectangle englobant
— enveloppe convexe

4 Le format GeoJSON

Les éditeurs de SIG proposent quasiment tous des formats propriétaires. Ces formats peuvent être *ouverts* (pensons aux shapefiles d'Esri) ou *fermés* (les géodatabases d'Esri). Par ouverts/fermés nous entendons que les spécifications de ces formats sont respectivement publiques ou pas. Pour l'utilisateur des données dans un format ouvert signifient des données qui pourront être ouvertes dans différents logiciels.

Il en sera de même pour le développeur : un format ouvert sera manipulable au travers de différentes librairies. Les formats fermés ne le seront généralement que via les librairies de l'éditeur de SIG propriétaire du format.

Aujourd'hui les données ont de plus en plus besoin d'être échangées entre divers acteurs. On comprendra alors aisément que l'importance des formats ouverts dans ce contexte. Parmi eux le **GeoJSON** semble aujourd'hui être plébiscité par la communauté SIG pour manipuler les données vecteur. Basé sur le format JSON (JavaScript Open Notation), le GeoJSON permet de représenter toutes les primitives géométriques définies par l'OGC. Il est lisible par un humain et manipulable par la plupart des libraires géospatiales.

1. voir les spécifications de l'OGC pour le détail complet de la validité des géométriques : <>

Le format JSON (JavaScript Open Notation) est un format textuel qui permet de représenter une information structurée. Il est fréquemment utilisé comme format d'échange dans le monde du web en raison de sa facilité de mise en oeuvre. Un document JSON ne comporte que deux types d'éléments : des couples clé/valeur et des listes ordonnées de valeurs. Les types permis pour les valeurs sont : nombres, chaînes de caractères, booléens, null, liste ordonnées ou objet JSON.

KML, GML, WKT : d'autres formats présentant les mêmes caractéristiques que le GeoJSON.

5 Le protocole `__geo_interface__`

Sean Gillies a proposé un protocole pour représenter les informations géographiques vecteur dans Python en se basant sur le GeoJSON. Ce protocole est adopté petit à petit par la communauté et les les principales libraires l'implémentent maintenant (`shapely`, `arcpy`, `geojson`, `PySAL`, etc.).

https://gist.github.com/sgillies/2217756

6 Exercice

enveloppe autour de points

7 Annexes

7.1 Principales libraires Python

— GDAL –> Fundamental package for processing vector and raster data formats (many modules below depend on this). Used for raster processing.
— Geopandas –> Working with geospatial data in Python made easier, combines the capabilities of pandas and shapely.
— Shapely –> Python package for manipulation and analysis of planar geometric objects (based on widely deployed GEOS).
— Fiona –> Reading and writing spatial data (alternative for geopandas).
— Pyproj –> Performs cartographic transformations and geodetic computations (based on PROJ.4).
— Pysal –> Library of spatial analysis functions written in Python.
— Geopy –> Geocoding library : coordinates to address <-> address to coordinates.
— GeoViews –> Interactive Maps for the web.
— Geoplot –> High-level geospatial data visualization library for Python.

— Dash –> Dash is a Python framework for building analytical web applications.
— OSMnx –> Python for street networks. Retrieve, construct, analyze, and visualize street networks from OpenStreetMap
— Networkx –> Network analysis and routing in Python (e.g. Dijkstra and A* -algorithms), see this post.
— Cartopy –> Make drawing maps for data analysis and visualisation as easy as possible.
— Scipy.spatial –> Spatial algorithms and data structures.
— Rtree –> Spatial indexing for Python for quick spatial lookups.
— Rasterio –> Clean and fast and geospatial raster I/O for Python.
— RSGISLib –> Remote Sensing and GIS Software Library for Python.

8 Bibliographie

— https://www.packtpub.com/application-development/python-geospatial-development-second-editio