Unidad 05
Lectura y escritura de datos geográficos

Curso de geoprocesamiento de datos con Python
2016

Cayetano Benavent Viñuales
Analista GIS en Geographica
cayetano.benavent@geographica.gs

Unidad 05 - Sumario de contenidos

1. Python y open source software para geoprocesamiento.
2. El Sistema de referencia de coordenadas: PROJ.4.
3. Manejo de fuentes de datos vectoriales: OGR.
4. Manejo de fuentes de datos ráster: GDAL.
5. Ejercicios prácticos.
6. Bibliografía

Python y open source software para geoprocesamiento

“What is geoprocessing?

Geoprocessing is a general term for manipulating spatial data, whether raster or vector. As you can imagine, that covers an awful lot of ground.

(...)

You even use geoprocessing in your daily life, whether you realize it or not.

(...)

Although some geoprocessing techniques are rather complicated, many are fairly simple. (...)”

(Garrard, 2016)

Python y open source software para geoprocesamiento

Python, desde hace ya más de una década, se ha configurado como el lenguaje favorito para geoprocesamiento de datos.

Los motivos de que Python se haya convertido en el lenguaje más usado para geoprocesamiento:

- Por todas las razones por las que Python es uno de los lenguajes favoritos en muchos ámbitos (ver unidad 01).

- Porque las librerías “core” para geoprocesamiento de datos están escritas en C/C++. Ello ha llevado a una integración con CPython fácil y directa.

Python y open source software para geoprocesamiento

En las unidades 05 y 06 veremos las principales librerías de geoprocesamiento:

PROJ.4, GDAL, GEOS

Conoceremos los binding Python de dichas librerías (Pyproj, Python-GDAL, Shapely, Fiona y Rasterio).

Veremos también las relaciones de estas librerías con el Stack de computación científica de Python (especialmente Numpy).

El Sistema de referencia de coordenadas: PROJ.4

PROJ.4 = Cartographic Projections Library

- Librería base para manejar Sistemas de Referencia de Coordenadas (CRS).

- Permite tanto conversiones entre distintas proyecciones cartográficas, como cálculos y transformaciones geodésicas.

- Una parte muy importante del geo-open source software (y mucho no open) utilizan Proj.4: QGIS, GRASS GIS, MapServer, PostGIS, Mapnik, GDAL, etc.

El Sistema de referencia de coordenadas: PROJ.4

- Esta librería ha sido liberada bajo la MIT license. La web oficial del proyecto: https://github.com/OSGeo/proj.4/wiki

- Está íntegramente escrita en el lenguaje C.

- Puede usarse a través de la API C o a través de su CLI.

- Esta librería fue desarrollada inicialmente (desde principios de los años 80) por G. Evenden en el USGS. Tras un breve periodo de inactividad, F. Warmerdam retoma los desarrollos en 2000, incorporndo con el tiempo la librería a la OSGeo (Open Source Geospatial Foundation - http://www.osgeo.org/).

El Sistema de referencia de coordenadas: PROJ.4

- Para manejar Proj.4 directamente en **Python**, existe la librería Pyproj, escrita por J. Whitaker (NOAA – ESRL).

- La librería tiene **dos grandes partes**:

PROJ: conversiones entre proyecciones cartográficas.
GEOD: para efectuar cálculos geodésicos.

El Sistema de referencia de coordenadas: PROJ.4

- Es muy importante saber la **versión** que estamos usando. La algoritmia del módulo de Geodesia cambia radicalmente entre las versiones 4.8 y 4.9.

- La versión 4.9 utiliza un port a C del módulo Geodesic de la librería GeographicLib de C. Karney. Es una librería muy sólida y precisa.

- Una breve discusión y una comparativa (desde Python) entre versiones de Proj4 y otras librerías, en un pequeño benchmark que escribí hace algún tiempo:

  https://github.com/cayetanobv/GeodeticMusings

El Sistema de referencia de coordenadas: PROJ.4

Para entender mejor el manejo de sistemas de referencia de coordenadas con Proj.4, realizaremos un ejercicio práctico siguiendo el Notebook unit05_01.

El Sistema de referencia de coordenadas: PROJ.4

Para entender mejor del módulo Geod de Proj.4, realizaremos un ejercicio práctico calculando líneas geodésicas (ver mapa abajo) siguiendo el Notebook unit05_02.

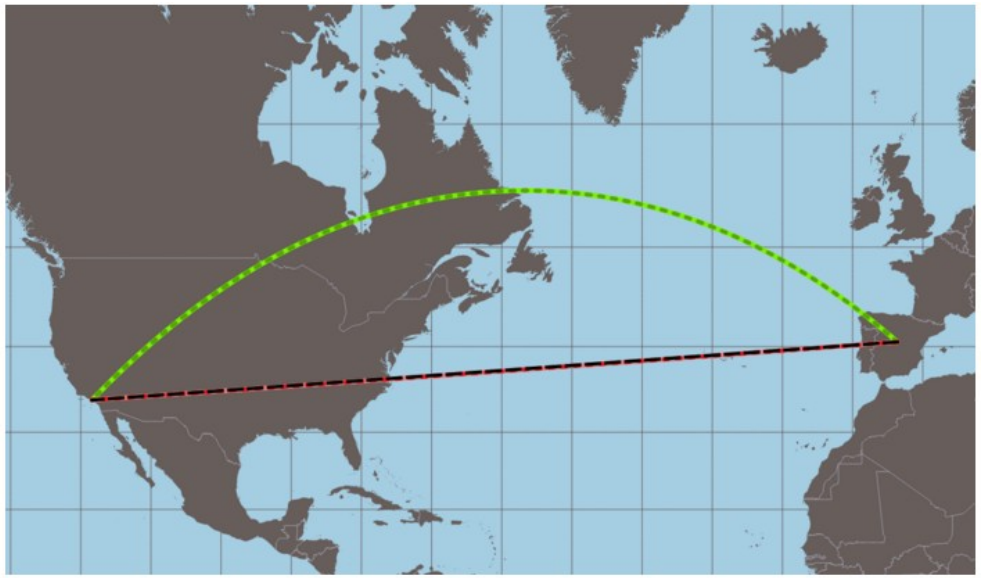


Fig 1. Mercator Projection.

Fig 4. Geodesic lines crossing 180º, Mercator Projection (Centered on 150ºE).

GDAL - Geospatial Data Abstraction Library

- Leer y escribir datos GIS espaciales, en el mundo del software libre, es casi sinónimo de trabajar con GDAL: http://www.gdal.org/

- Desde la v1.3.2, GDAL y OGR se unen en un mismo proyecto (pertenecen al mismo “source tree”), y pasa a llamarse GDAL.

- GDAL es la capa de lectura/escritura de casi todo el software libre geo (¡y de mucho software no libre!):
  https://trac.osgeo.org/gdal/wiki/SoftwareUsingGdal

Leer y escribir datos espaciales: GDAL

- GDAL se distribuye bajo una licencia de estilo MIT/X:
  https://trac.osgeo.org/gdal/wiki/FAQGeneral#WhatlicensedoesGDALOGRuse

- Está escrita en C/C++, y su API principal, obviamente, se accede desde C/C++.

- Hay APIs en otros lenguajes. Las activas actualmente son: C#, Java, Perl, and Python.

- Otras APIs a GDAL externas al proyecto (“outside of the GDAL source tree”): Go, Lua, Node.js, R, Tcl, and VB6.

Leer y escribir datos espaciales: GDAL

- GDAL soporta actualmente (abril 2016) unos 142 formatos raster y 84 vectoriales:

  raster--> http://www.gdal.org/formats_list.html

  vectorial --> http://www.gdal.org/ogr_formats.html

- GDAL permite escritura en una gran parte de los formatos soportados (no todos).

- Muchos formatos requieren compilación específica de ciertos drivers (o librerías dependientes).

- Con cada versión nueva, se añaden multitud de formatos y nuevas capacidades (¡revisar bien CHANGELOG!).

Leer y escribir datos espaciales: GDAL

- Con la versión actualmente estable (GDAL 2.0), la API cambia por completo. Su integración con OGR es ya muy elevada.

- Hay que tener en cuenta que la mayor parte del software libre que depende de GDAL para leer o escribir (casi todo!!), aún está en proceso de adptación al cambio de v1.X a v2.0.

- Para ver un resumen con las principales novedades, recomiendo leer la presentación de Even Roualt (actual PSC Chair) del FOSS4Geo2015:

http://download.osgeo.org/gdal/presentations/GDAL%202.0%20overview%20(FOSS4G-E%202015).pdf

Leer y escribir datos espaciales: GDAL

- Aunque abundante, la información sobre GDAL es farragosa de entender para usuarios no expertos del mundo GIS. La librería funciona a la perfección, pero es algo “barroca”.

- Recomiendo por ello, acceder a lecturas externas más didácticas, como el Cookbook de Jared Erickson:
  http://pcjericks.github.io/py-gdalogr-cookbook/index.html
- y los cursos de Chris Garrard: http://www.gis.usu.edu/~chrisg/python/2009/
  Nota: Las últimas son un clásico, pero están algo desactualizadas. C. Garrard está a punto de publicar un libro: https://www.manning.com/books/geoprocessing-with-python

- En cualquier caso, hay que estar atentos, porque la v2.1 trae novedades cruciales:
  http://erouault.blogspot.com.es/2015/10/gdal-and-ogr-utilities-as-library.html

Leer y escribir datos espaciales: GDAL

- Antes de empezar a trabajar con GDAL, siempre debemos asegurarnos de dos cosas:

1. Versión de GDAL instalada:

```
$ gdalinfo --version
```

2. Formatos soportados en la versión que tenemos instalada:

```
$ gdalinfo --formats
```

Ver detalles de formatos concretos:

```
$ gdalinfo --format GTIFF
```

Leer y escribir datos espaciales: GDAL

- La API de Python es una de las más utilizadas. La wiki oficial es:

  https://trac.osgeo.org/gdal/wiki/GdalOgrInPython

- GDAL suele ser la primera capa de cualquier desarrollo o flujo de geoprocesamiento que diseñemos para trabajar con datos GIS.

- La documentación de la API (generada automáticamente):

  http://gdal.org/python/

Leer y escribir datos espaciales: GDAL

- GDAL provee un conjunto de utilidades de línea de comando muy potentes:
  http://www.gdal.org/gdal_utilities.html

- Estas utilidades, escritas en C/C++ o en Python, son excelentes ejemplos de cómo manejar la API de GDAL.

- Los test de GDAL están casi todos escritos en Python, por lo que estudiarlos es otra manera de aprender:
  https://svn.osgeo.org/gdal/trunk/autotest/

Leer y escribir datos espaciales: GDAL

• No obstante lo potente de GDAL, es importante tener en cuenta que hay numerosos “Gotchas” a tener en cuenta si usamos lo usamos desde Python.

• Ejemplos (algunos sorprendentes para programadores expertos en Python):

- Python bindings do not raise exceptions unless you explicitly call UseExceptions().

- Certain objects contain a Destroy() method, but you should never use it.

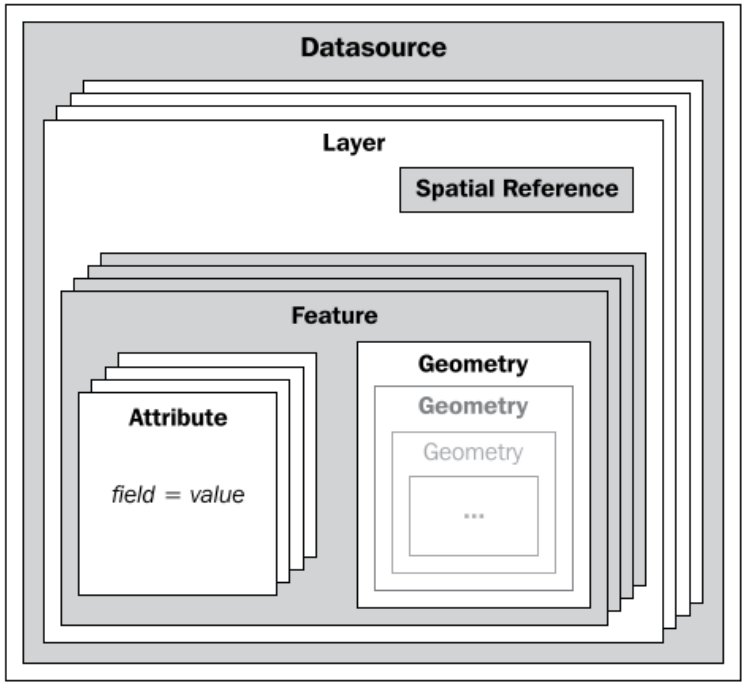
- etc.

https://trac.osgeo.org/gdal/wiki/PythonGotchas

Leer y escribir datos vectoriales: GDAL

OGR es la parte de GDAL encargada de gestionar los formatos vectoriales. Aprendamos cómo se estructura el modelo de datos vectorial en OGR:

(Detalles en http://www.gdal.org/ogr_arch.html)



(E. Westra, 2014)

Leer y escribir datos vectoriales: GDAL

Para entender mejor cómo se leen datos vectoriales con OGR, realizaremos un ejercicio práctico siguiendo el Notebook unit05_03.

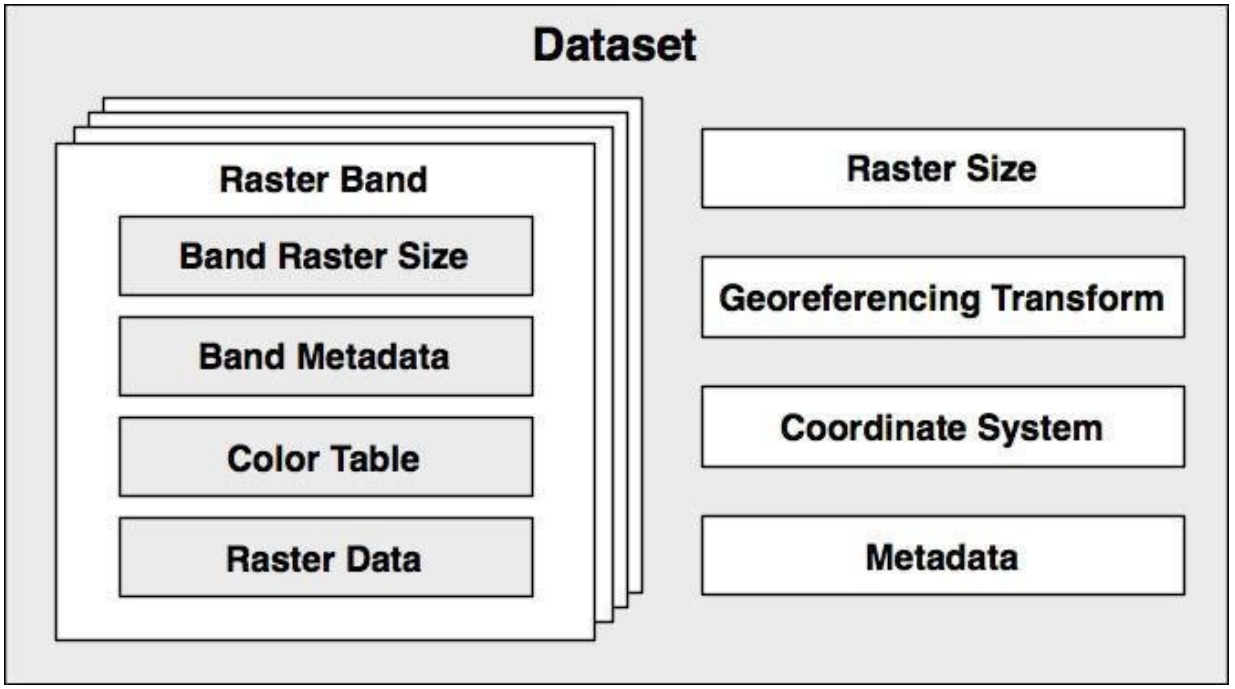
Leer y escribir datos vectoriales: GDAL

Para entender mejor cómo se escriben datos vectoriales con OGR, realizaremos un ejercicio práctico siguiendo el Notebook unit05_04.

Leer y escribir datos raster: GDAL

Aprendamos ahora a leer y escribir datos raster desde GDAL. En primer lugar, debemos conocer cómo se estructura el modelo de datos raster en GDAL:

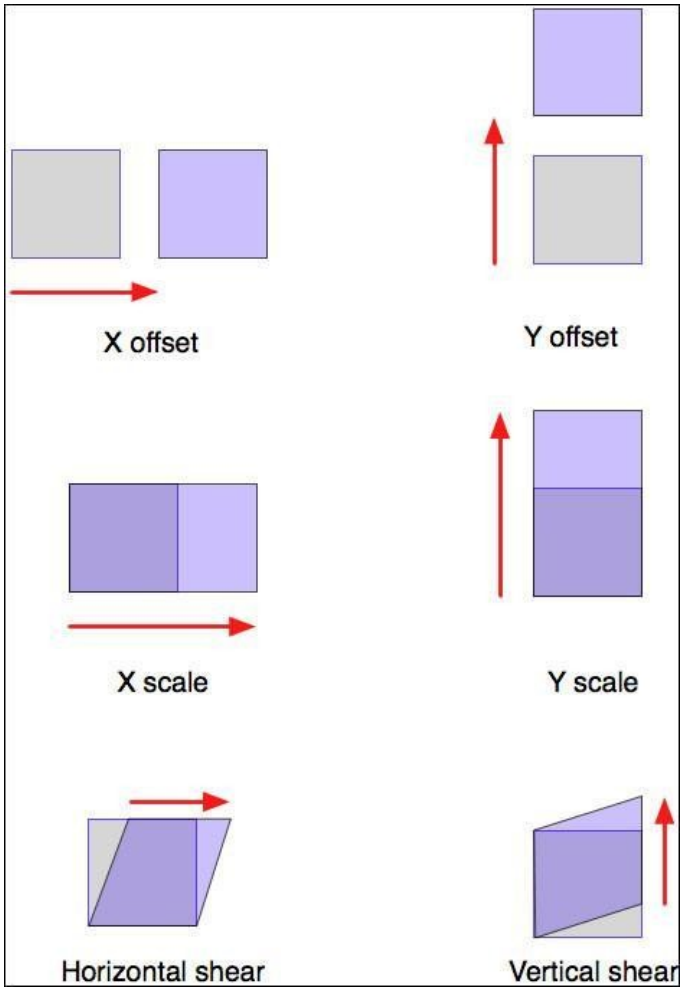
(Detalles en http://www.gdal.org/gdal_datamodel.html)



(E. Westra, 2014)

Leer y escribir datos raster: GDAL

- La geotransformación se construye con los siguientes parámetros:



(E. Westra, 2014)

Leer y escribir datos raster: GDAL

Para entender mejor la lectura de datos raster con GDAL, realizaremos un ejercicio práctico siguiendo el Notebook unit05_05.

Leer y escribir datos raster: GDAL

Para entender mejor la escritura de datos raster con GDAL, realizaremos un ejercicio práctico siguiendo el Notebook unit05_06.

Leer y escribir datos espaciales con Fiona y Rasterio

Rasterio y Fiona son una excelente alternativa para leer y escribir datos raster y vectoriales desde Python.

No son un wrapper al binding de Python. Están construidos sobre GDAL con Cython.

Es más sencilla de manejar que el binding “oficial”, más rápida y mucho más Pythónica. No alcanzan el 100% de funcionalidad de GDAL, pero solventan de una manera muy eficiente casi todo lo que se usa habitualmente.

Leer y escribir datos espaciales con Fiona y Rasterio

Tanto Rasterio como Fiona, además de como librería Python, incluyen una interfaz de línea de comando (rio y fio):

```
$ rio info data/mde/h10_1050_2-2/h10_1050_2-2.tif
```

```
$ fio info ../data/geologia_marina/mm01_geologia.shp
```

Hay una alternativa paralelizada para Rasterio (Rio-mucho):

```
$ rio-mucho
```

Leer y escribir datos espaciales con Fiona y Rasterio

- Lecturas interesantes para aprender Rasterio y Fiona:

http://sgillies.github.io/foss4g-2014-fiona-rasterio

https://github.com/sgillies/frs-cheat-sheet

“- A Python package at the top.
- Extension modules (using Cython) in the middle.
- Fast loops, typed memoryviews, "nogil" blocks.
- GDAL shared library on the bottom.”

Leer y escribir datos espaciales con Fiona y Rasterio

Para entender mejor la lectura de datos vectoriales con Fiona, realizaremos un ejercicio práctico siguiendo el Notebook unit05_07.

Leer y escribir datos espaciales con Fiona y Rasterio

Para entender mejor la escritura de datos vectoriales con Fiona, realizaremos un ejercicio práctico siguiendo el Notebook unit05_08.

Leer y escribir datos espaciales con Fiona y Rasterio

Para entender mejor el manejo de datos ráster con Raster, realizaremos un ejercicio práctico siguiendo el Notebook unit05_04.

Bibliografía

Selección bibliográfica:

- Westra, Eric (2013): “Python Geospatial Development (Second Edition)”. Packt publishing.

- Garrard, Chris (2016): “Geoprocessing with Python”. Manning.