Ghanaian Crop Modelling Jupyter Notebook and Data Cube

Cheap & Cheerful Geospatial Dissemination

J Gómez-Dans

National Centre for Earth Observation (NCEO)
Department of Geography
University College London

Background

Monitoring agriculture in West Africa

- Our aim is to deploy systems that monitor crop condition using data assimilation (DA) techniques that blend EO data and mechanistic crop growth models.
  - For smallholder farming, data such as Sentinel 1 and 2, Landsat are required.
  - . . . but crop models & interpretation also require meteo data.
  - . . . and often things like MODIS etc.
  - Multi-temporal analysis is fundamental: we work on data stacks, not images.
  - Masses of data.
  - ”Cheap & cheerful”, low maintenance, shoestring budget for dissemination.

The users

- Working with partners in developing countries can be challenging:
  - Limited infrastructure to do big processing and/or storage
  - Limited network connectivity
- Common challenges to multidisciplinary research
  - Broad range of software familiarity (not everyone speaks Python!)
  - Broad distribution of user backgrounds (agronomists, meteorologists, EO folk, . . .)
- Minimise effort in maintaining system.
- Aim is to have portable systems that can be deployed by partners.

Types of users

Who's going to use the data? Two main types

1. Casual users (e.g. workshops, people trying things out, . . .).
   - Download small(ish) data & process locally on e.g. laptop.
2. Dedicated users (e.g. people producing e.g. regional monitoring reports).
   - Work on some remote cloud infrastructure, process big chunks.

Data requirements

As an user, what do we want from the data?
A lot of inspiration comes from Google Earth Engine.

- Easy to subset spatially, temporally & thematically (spectrally, parameter, etc).
- Make everything a continuum: large area lightweight mosaics
- Ensure data can be overlapped e.g.
  - Meteo
  - Land Use
  - EO data (e.g. LAI, fAPAR, σ^0, ...)

Some assumptions

- Internet cable available somewhere.
- *Assume* users have processing needs, not just RGB visualisation.
- Assume a minimum level of technical competence.
- Use well-known, widely available libraries.
- Easy path for user casual to determined user migration.
- Portable, easy to maintain system.
- System should be easily extensible with minimum fuss.

Observations on geospatial data access and processing

Data dissemination

- Includes common pre-processing:
  - E.g. atmospheric correction, parameter retrieval, ...
  - Auto-update (e.g. download from sentinel hub, Copernicus, etc.)
- Exploit HTTP transport.
  - Available everywhere
- Users can request arbitrary space/time/parameter subsets
- Allow both direct usage in e.g. notebooks and also download to local.
- Allow local processing for dedicated users (no HTTP overhead)

Design choices

HTTP Make everything available via stateless HTTP

- Ubiquitous
- Mature technology
- Ample experience in load balancing
- Simple to set up and maintain

Jupyter notebooks Ideal for data access and analysis

- Low bandwidth
- Latency tolerance
- Familiarity
- Access to remote HPC facilities

Python As a main language

- Familiarity
- Ecosystem of packages
- However, open to other communities (e.g. R)

Implementation

The Data Cube (*sic*)

- Serving Geo data $\longrightarrow$ serving chunks of files via HTTP server.
- GDAL provides a transparent HTTP transport for many formats.
- Data stored to allow seeking spatial blocks $\Longrightarrow$ Cloud Optimized GeoTIFF (or GTiff in chunks)
- Scaling is left to web server, cacheing, replication, etc. 20+ years experience on this!
- A catalogue that maps e.g. dates to GeoTIFF URLs

Store mosaics in a suitable format in an HTTP accessible folder, and provide a JSON file that provides the URLs of data layers.

Centralised processing on JASMIN

We have used the JASMIN infrastructure.

- Lots of cores & fast disks.
- Lots of mirrored data already present
- Can share folders via HTTP.
- Can mount folders in virtual clusters
- It's free!

But easy to deploy elsewhere (e.g. using my own UCL webspace!). Strong requirement: a HTTP server with HTTP/2.0 capabilities

- Input data downloading etc.
- Exploit batch queuing system for pre-processing.
- Create ruf'n'ready catalogue.
- Use simple scripts and cronjobs to automate tasks.

$\Longrightarrow$ Minimal management requirements.

User experience

1. Read JSON file to find out file locations, etc.
2. Pass remote URLs to GDAL
3. Pass spatial subset parameters (GeoJSON or whatever) to GDAL
4. Asynchronous download to either:
   - Numpy arrays
   - Local files

Library $\longrightarrow$ `GomezEngine`.
But you can readily use `xarray` if you want.
Data available via QGIS

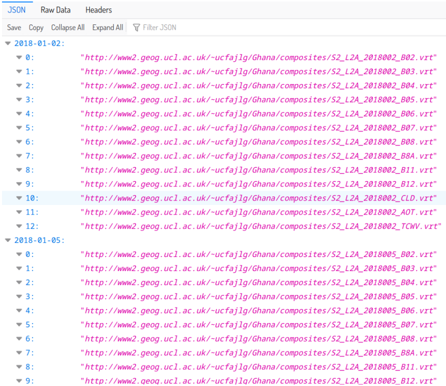

Figure: The JSON catalogue

National Centre for Earth Observation
NATURAL ENVIRONMENT RESEARCH COUNCIL

How does this work

Example (I)

```
[3]: ds = GomezEngine.DataStorageSentinel2("http://www2.geog.ucl.ac.uk/~ucfajlg/Ghana/database.json")
     print(f"Total number of acquisitions: {len(list(ds.data_db.keys())):d}")
     print(f"First date: {list(ds.data_db.keys())[0].strftime('%d %b %Y'):s}")
     print(f"Last date: {list(ds.data_db.keys())[-1].strftime('%d %b %Y'):s}")

     Total number of acquisitions: 50
     First date: 02 Jan 2018
     Last date: 28 Dec 2018
```

Figure: Accessing the catalogue

- Import library, and request catalogue
- The DB is just a python dictionary
- All ”layers” have same spatial extent

Example (II)

```
[8]: analysis_data = ds.extract_band(["B04", "B08"], roi="feature.geojson")
     for k in analysis_data['B04'].keys():
         print (k.strftime("%Y-%m-%d"))

100% 100/100 [01:01<00:00, 1.32it/s]

2018-02-11
2018-01-22
```

Figure: Grabbing the data

- Assume your region of interest is in a GeoJSON file
- . . . or a Python string, or shapefile, or whatever!
- Select selected layers (”bands”)
- Get a dictionary of already subset numpy arrays

Final remarks

- Focus on empowering our partners to develop and maintain their own infrastructures.
- Focus on limited resources
- Consider how the data will be used, not how the data is distributed.
- Simplify maintenance and routine processing.
- Very simple ”API” via exploiting GDAL's capabilities.
- Shoestring project started 1+ year ago:
  - Commercial and other offerings probably supersede the need for this.
  - Community is fast moving and hard to keep track of things.
  - KISS approaches have their interest.

Try it for yourself!

UCL

`http://bit.ly/2VDUFzn`

SCAN ME

National Centre for Earth Observation
NATURAL ENVIRONMENT RESEARCH COUNCIL