Working with `large` datasets

March 11, 2022

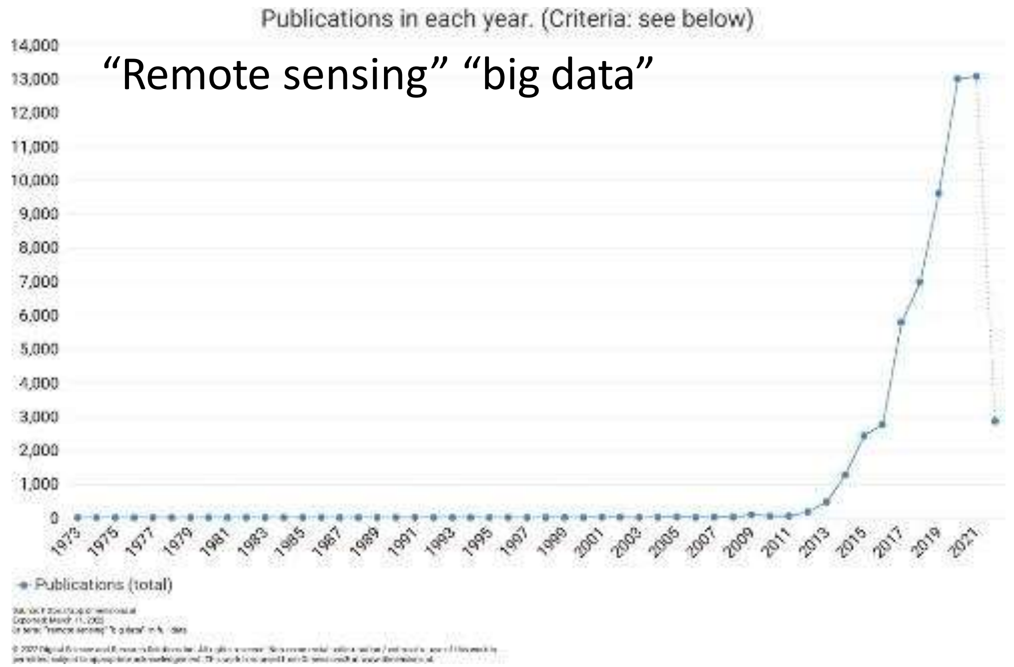
Alexandre.Bevington@gov.bc.ca

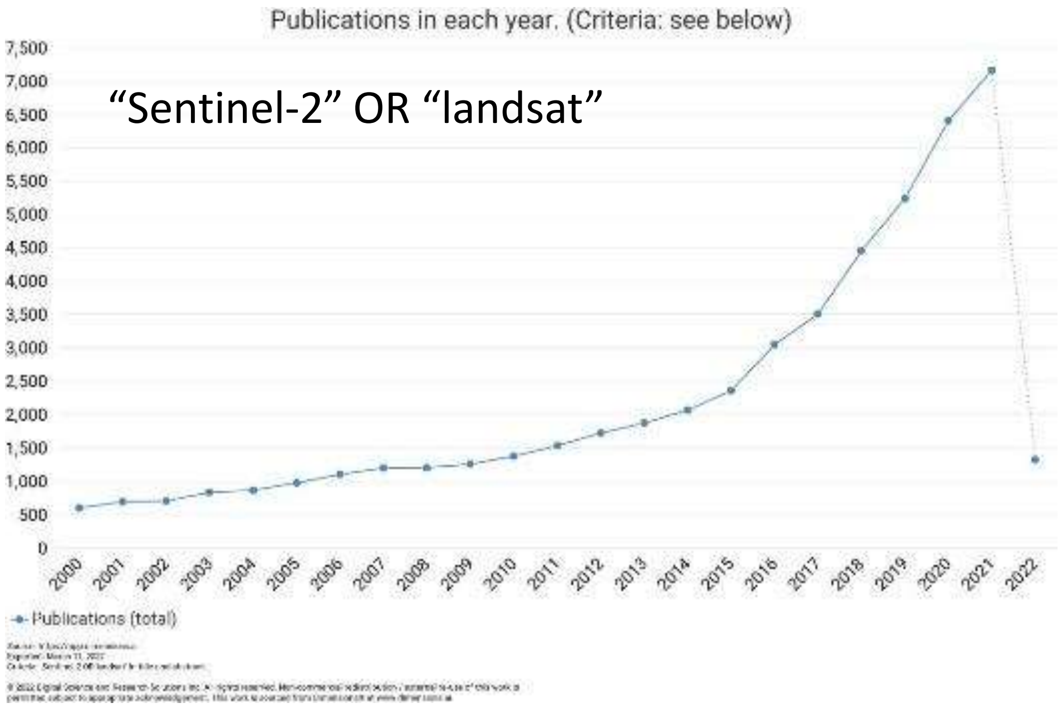
What are large datasets?

- More capabilities = Bigger questions

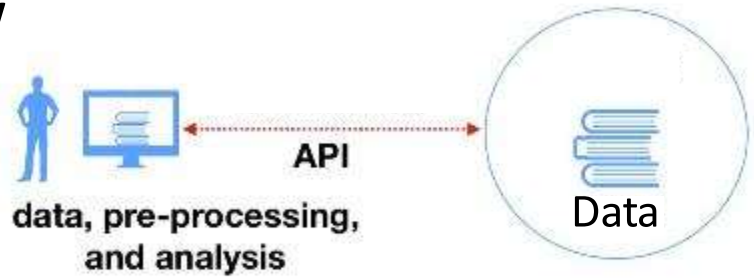
```
from sentinelsat import SentinelAPI
api = SentinelAPI('username','pw')
```

```
S1 = api.query(date=('2021-01-01T00:00:00Z','2021-12-31T23:59:59Z'),platformname='Sentinel-1',producttype='SLC',sensoroperationalmode='IW')

api.get_products_size(S1)

~ 2.07 Petabyte of Sentinel-1 IW SLC data (NOT INCLUDING GRD)

S2 = api.query(date=('2021-01-01T00:00:00Z','2021-12-31T23:59:59Z'),platformname='Sentinel-2',producttype='S2MSI1C')

api.get_products_size(S2)

~ 0.94 Petabyte of Sentinel-2 L1C dat
```

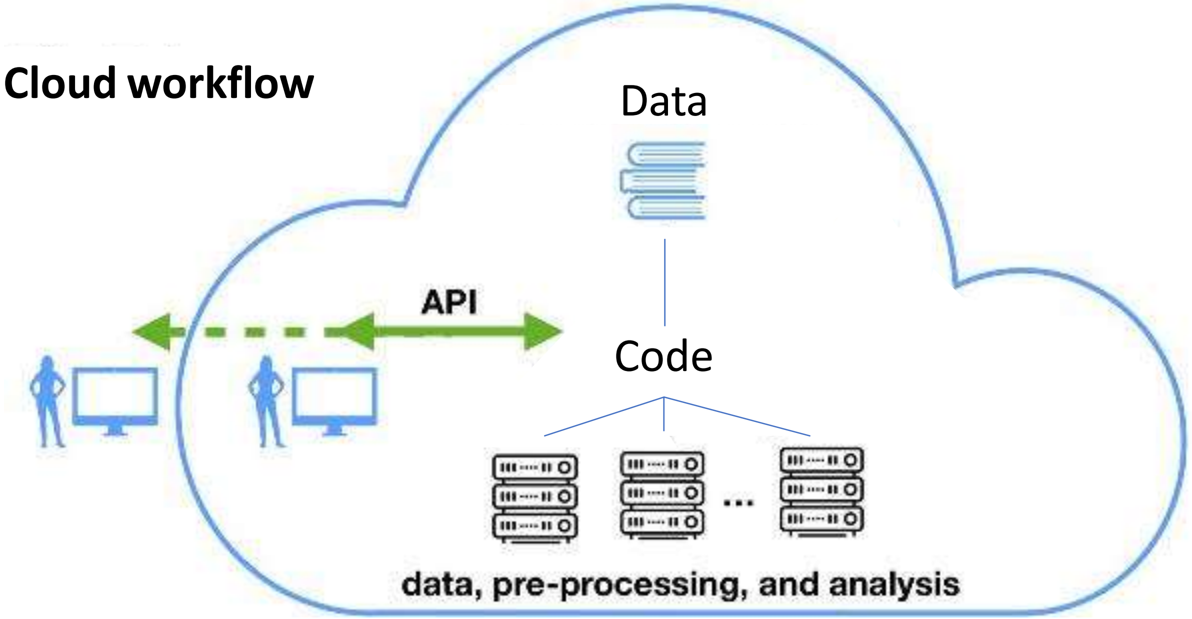




Local workflow



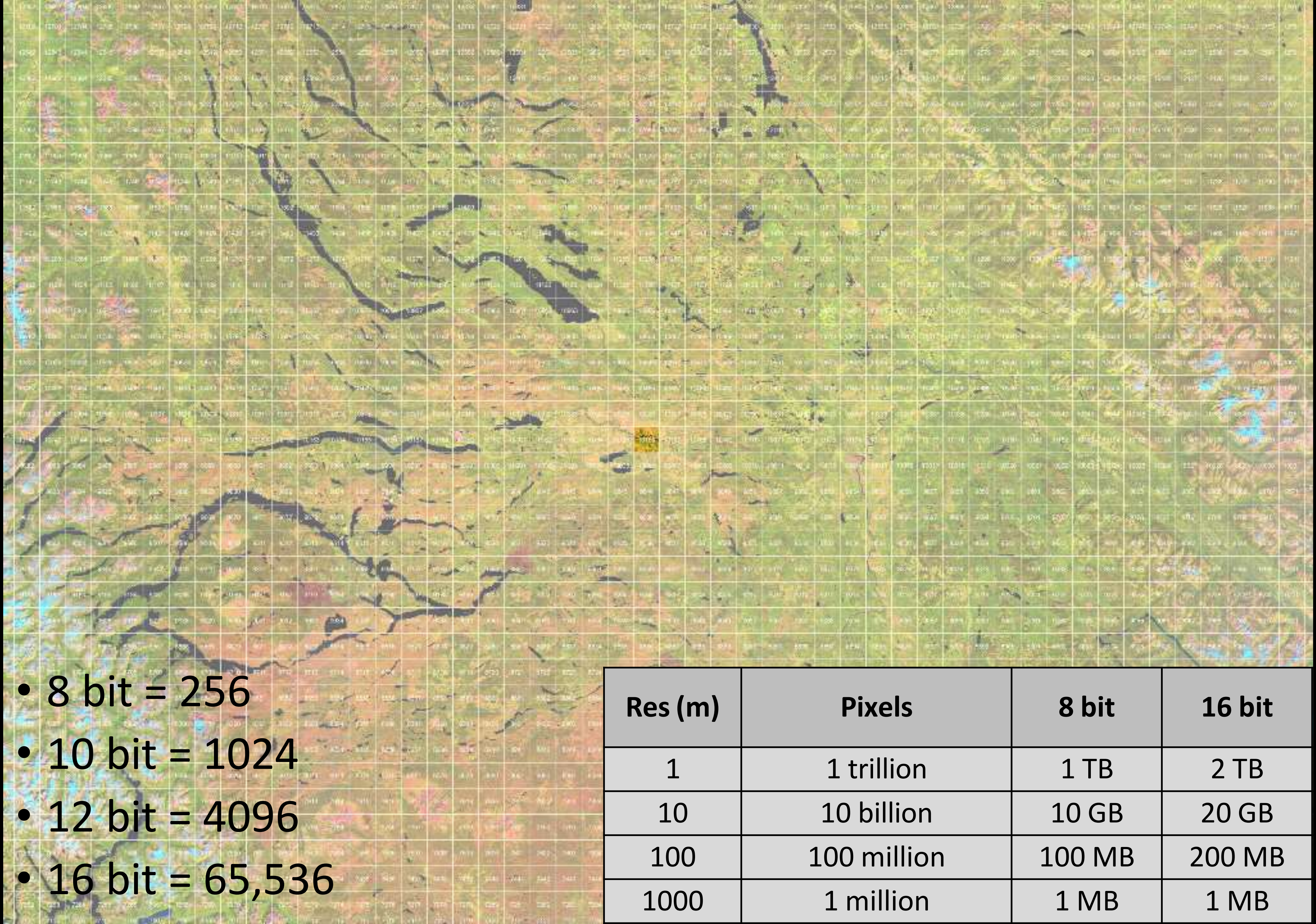
Cloud workflow



https://escience.washington.edu/uw-researchers-are-changing-the-way-we-use-satellite-observations-of-earth/

Computer speed

- **CPU – Central Processing Unit**
  Speed of processor, number of cores (How many workers)
- **RAM – Random Access Memory**
  Store working data and machine code (Multitasking)
- **SSD vs HDD – Solid State vs Hard Disk**
  Read/write speed and total storage
- **GPU - Graphics Processing Unit**
  Speeds up image visualization and processing, optimal for some tasks
- etc

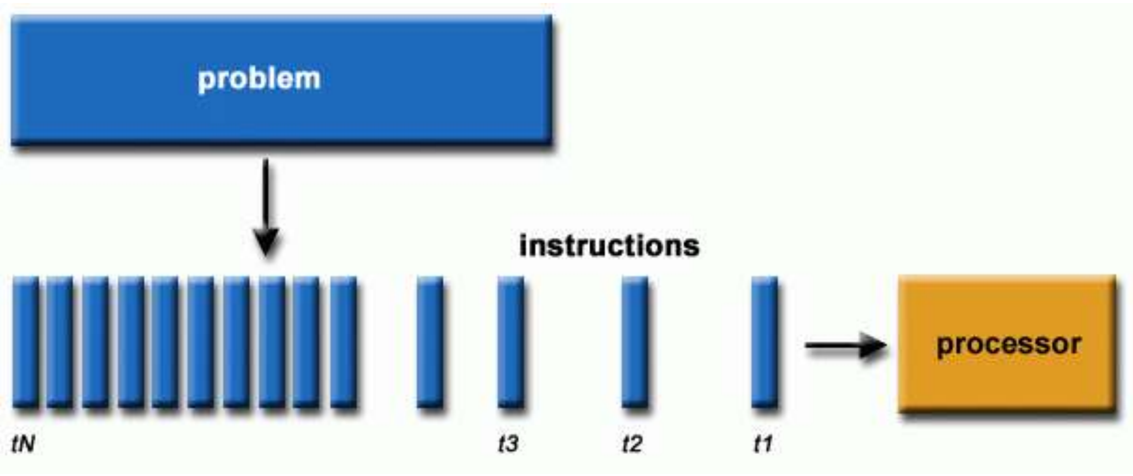


- 8 bit = 256
- 10 bit = 1024
- 12 bit = 4096
- 16 bit = 65,536

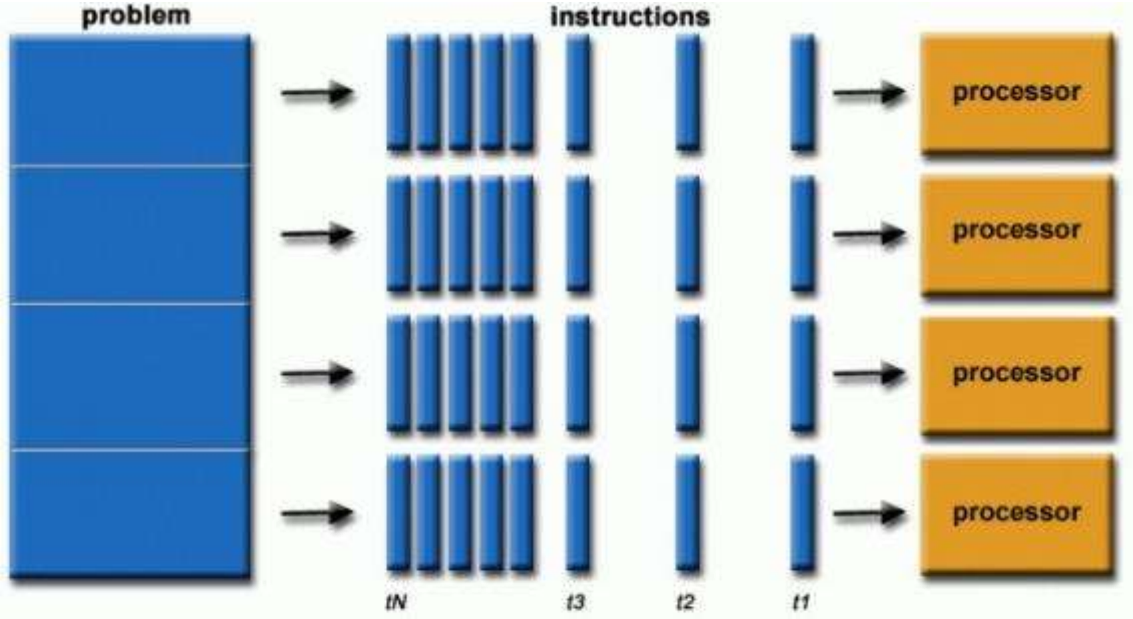
Res (m)	Pixels	8 bit	16 bit
1	1 trillion	1 TB	2 TB
10	10 billion	10 GB	20 GB
100	100 million	100 MB	200 MB
1000	1 million	1 MB	1 MB

- **Serial computing**
    - A problem is broken into instructions
    - Executed sequentially on a single processor
    - One instruction executed at a time
- **Parallel Computing**
    - A problem is broken into parts
    - Each part is broken into instructions
    - Execute simultaneously on different processors
    - Requires orchestration, sometimes not worth it
- **Hyper-threading**
    - Better task scheduling
    - Minimizes processor downtime
    - Works for both serial and parallel computing
    - Not equivalent to more cores

Serial computing



Parallel Computing



https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial

How to work in Parallel?

- Not all functions can be run in parallel
- SAGA GIS runs in parallel by default
- GRASS can be parallel with OpenMP
- Python: use `Dask`
- R: use `Future`
- ArcGIS: Available for some functions, not all

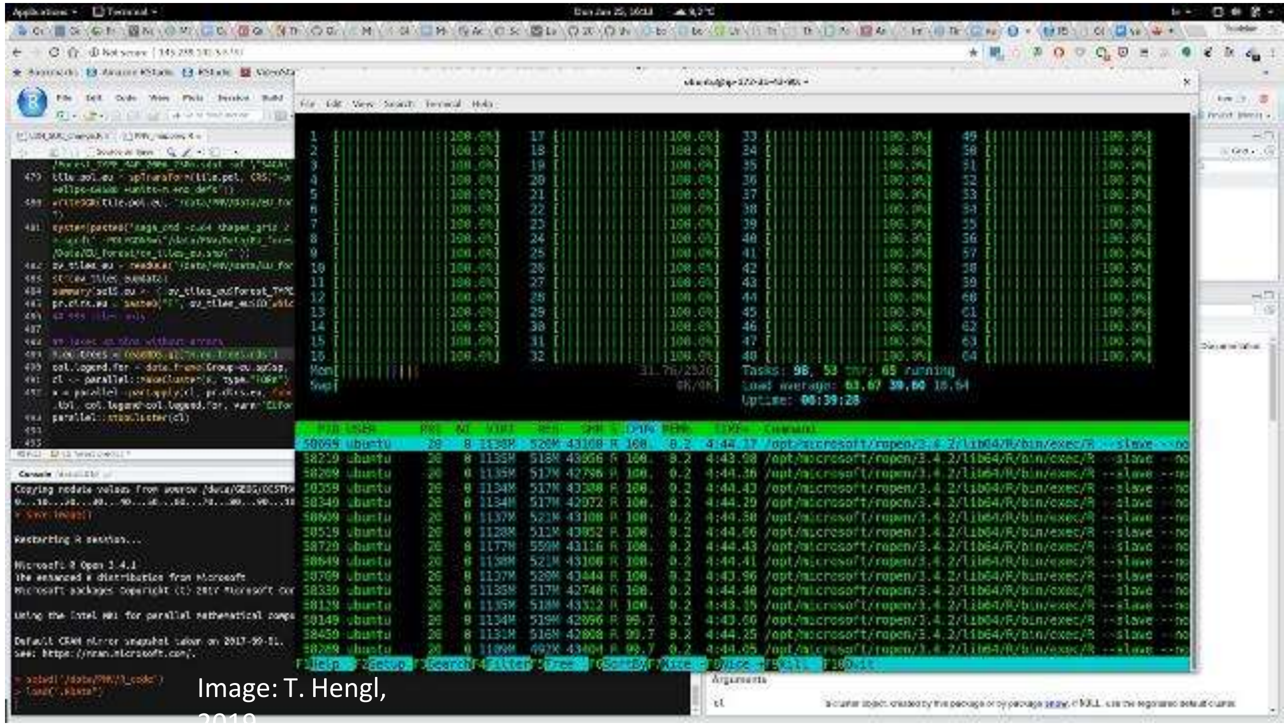
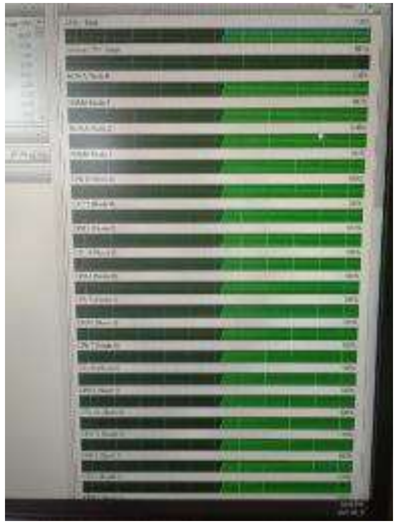
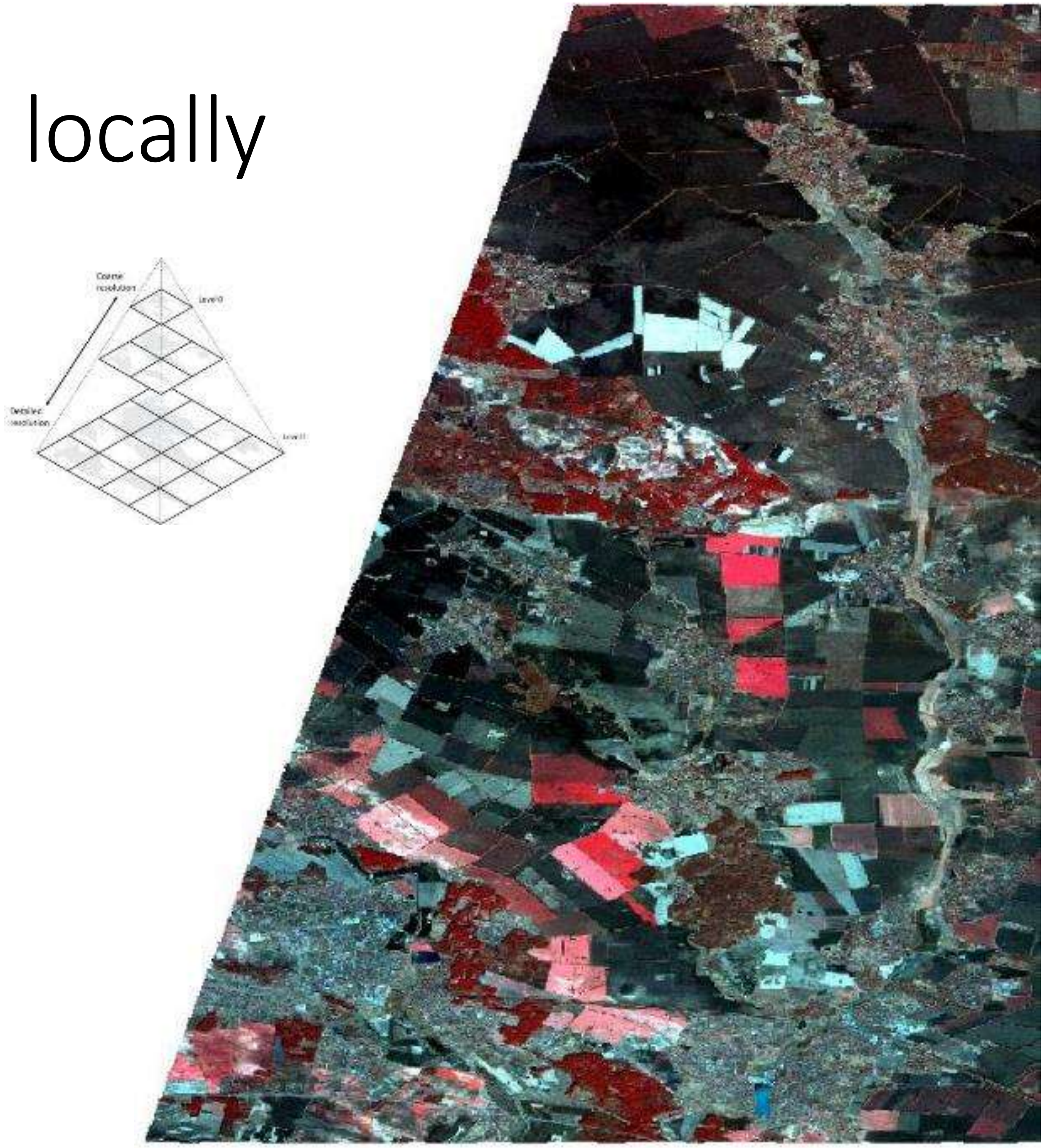


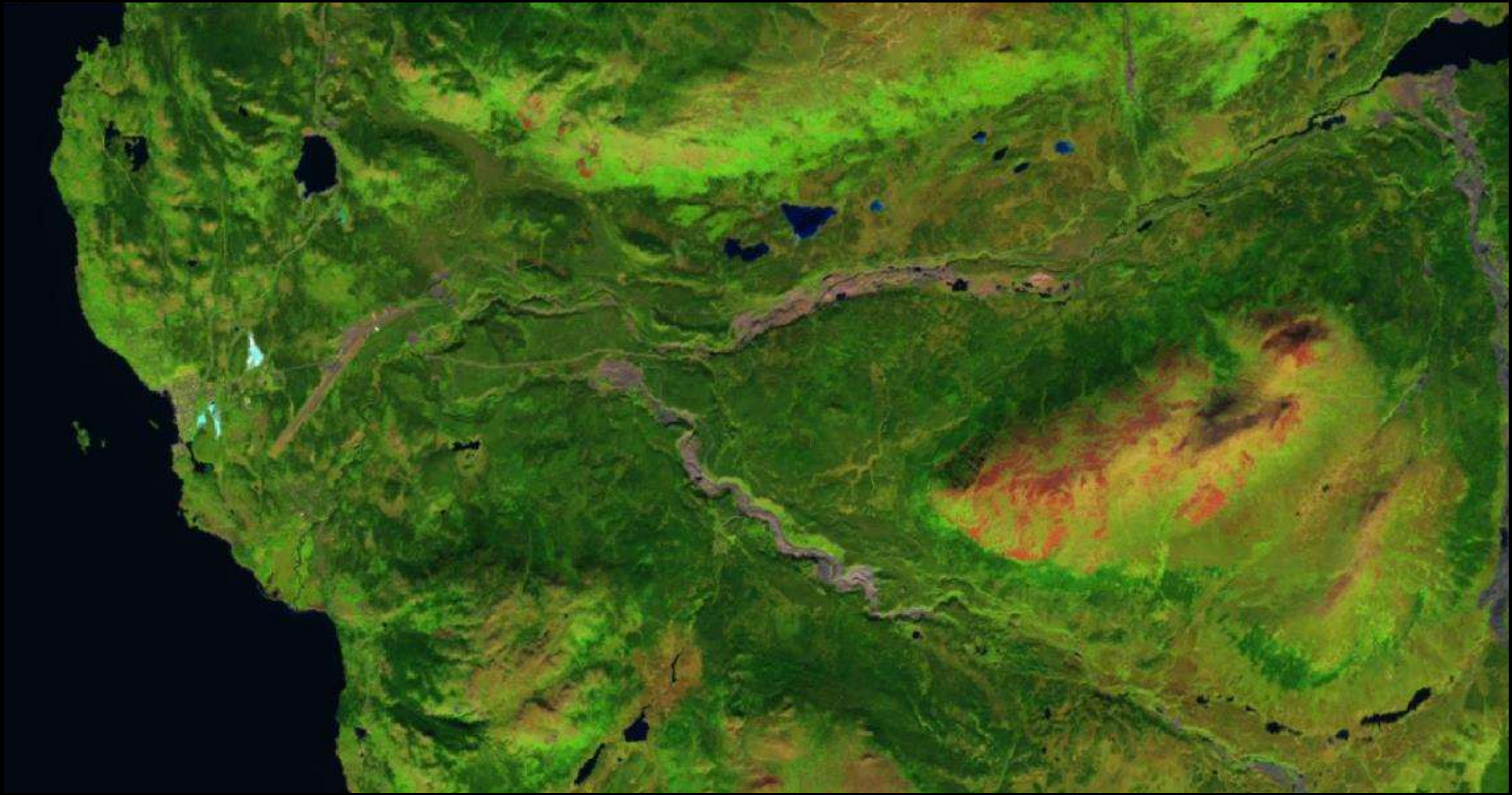
Image: T. Hengl,



Managing large images locally

- Tiles
  - Easier to manage as small tiles
- Visualize all tiles at once
  - **Virtual raster (gdalbuildvrt)**
    Creates index of all tiles (small file)
  - **Mosaic (gdalmerge)**
    Combines all images into a large file
- Speed up visualization
  - **Overviews (gdaladdo)**
    Creates multiple reduced resolution layers that are used at different zoom levels, speeds up visualization. Layers stored in *.ovr file

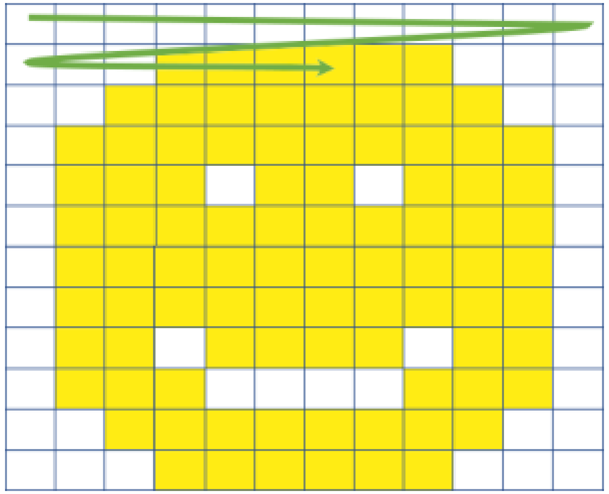




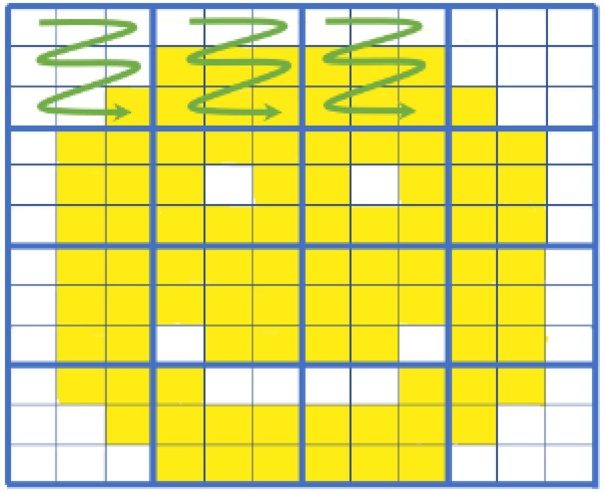
Cloud optimized geotiffs

- A COG is a regular GeoTIFF
- COGs have an internal organization that supports efficient access via HTTP GET range requests
- Supports overviews
- Clip rasters BEFORE processing downloading
- Used by:
  - STAC Index
  - Google Earth Engine
  - DigitalGlobe/Mazar
  - USGS
  - etc.

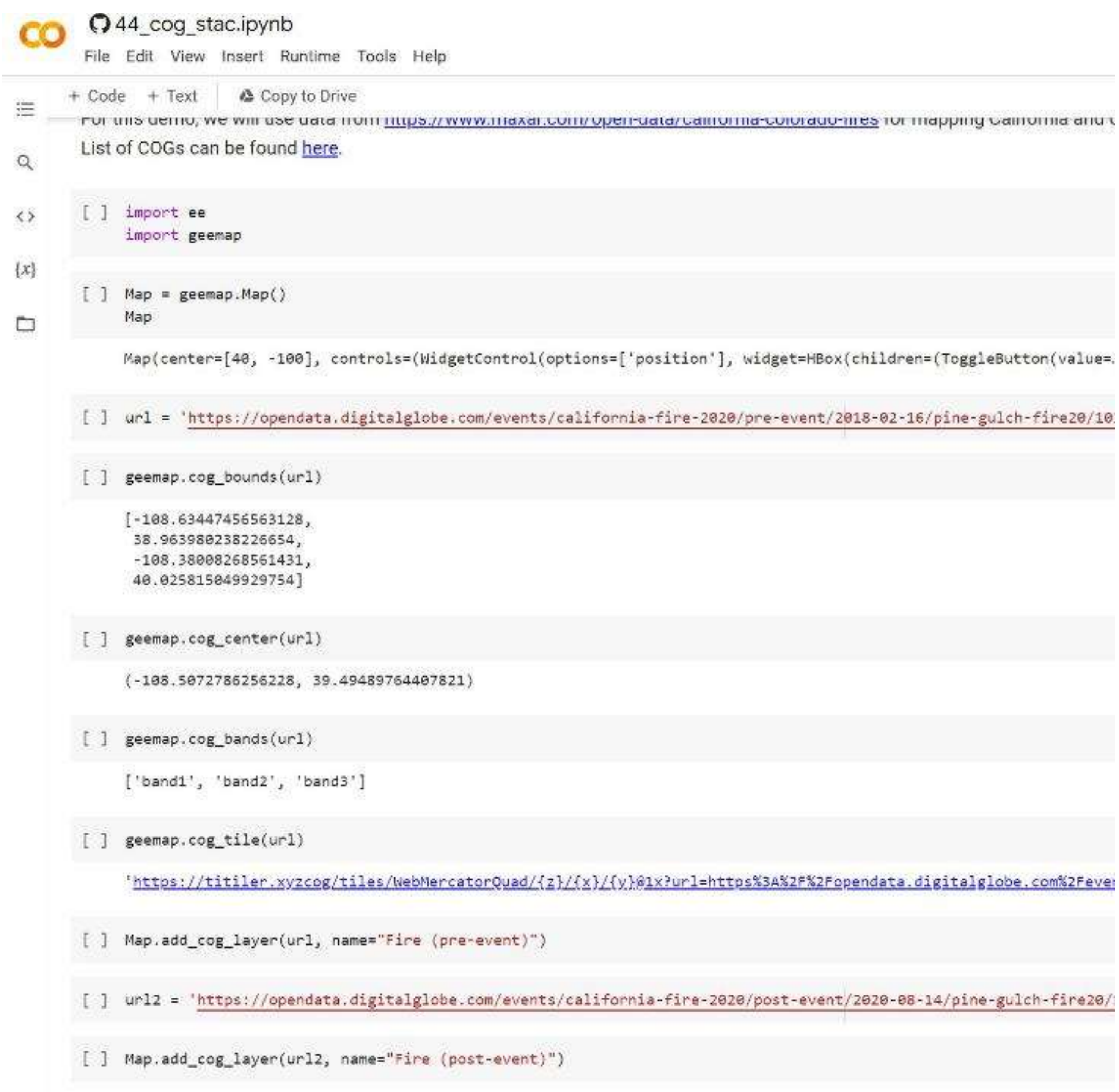
Typical



COG



https://www.element84.com/blog/cloud-optimized-geotiff-vs-the-meta-raster-format

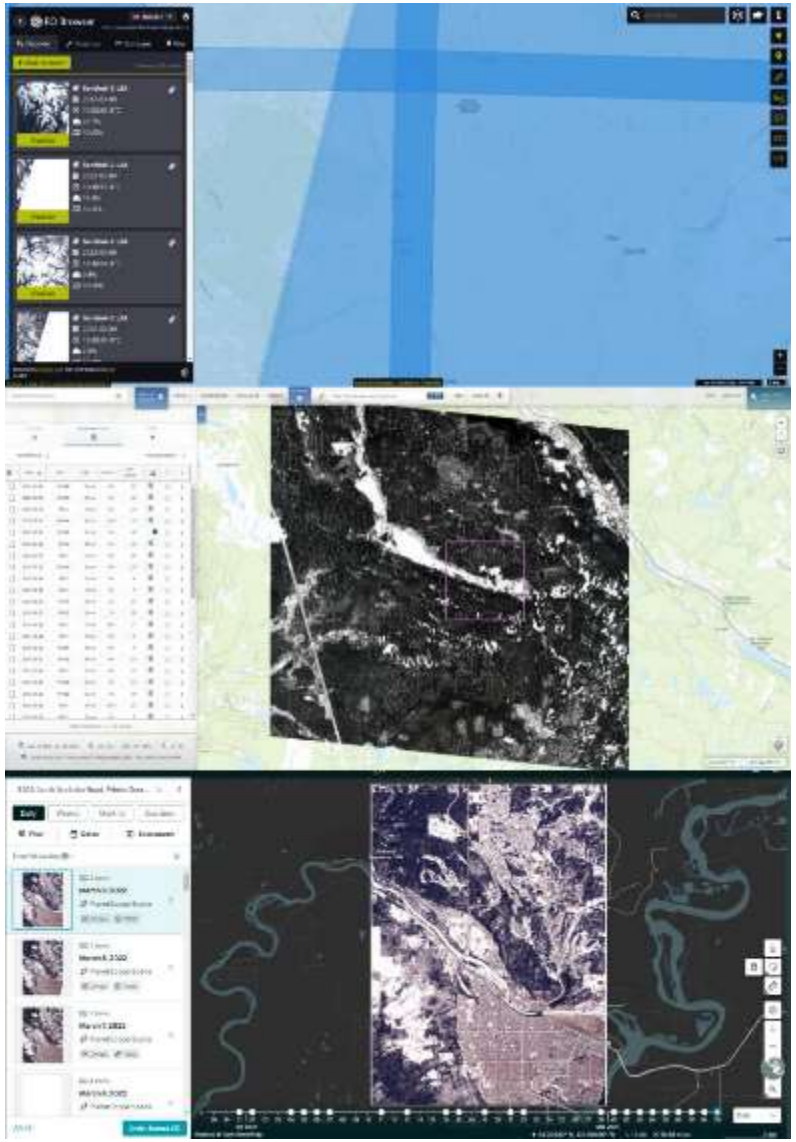


DEMO

https://colab.research.google.com/github/giswqs/geemap/blob/master/examples/notebooks/44_cog_stac.ipynb

Spatiotemporal Asset Catalog (STAC)

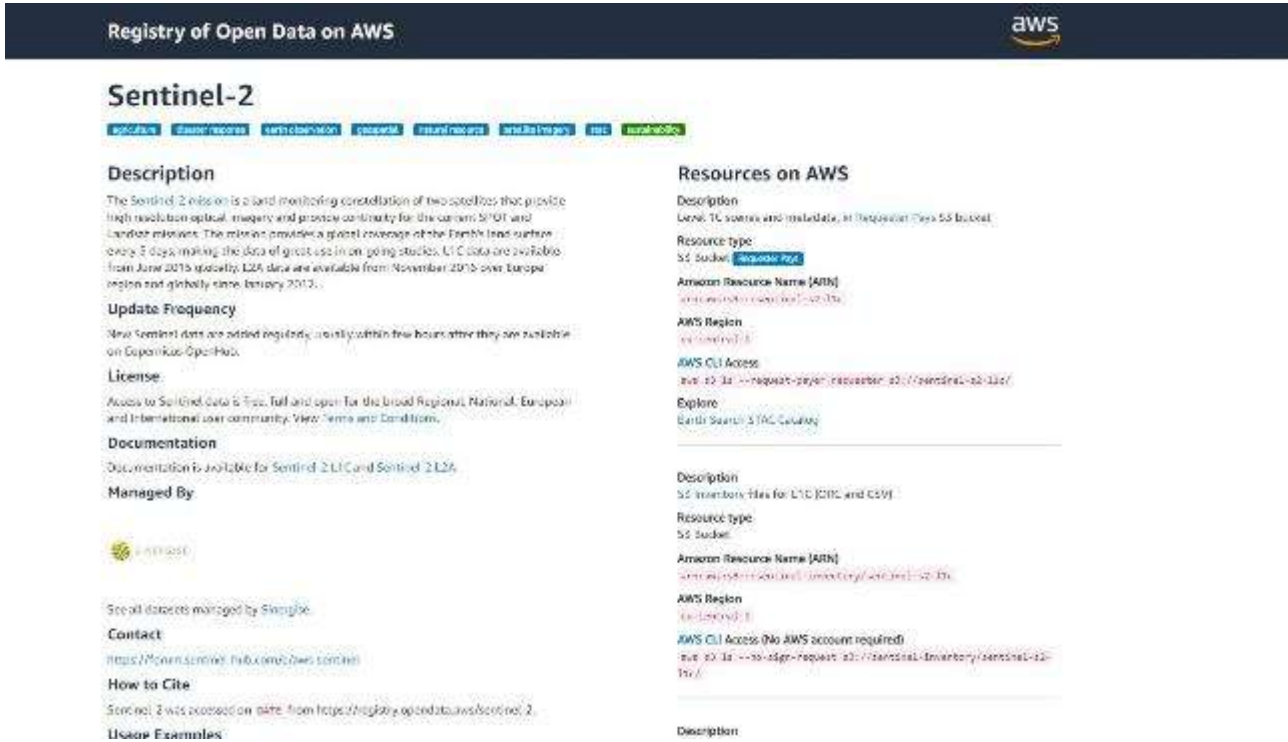
- Manage geospatial data with a single language
- Ideal for searching and managing datasets (not for processing or visualization)
- Built for the cloud using an open standard format for simple geographical features, along with their non-spatial attributes (GeoJSON)
- STAC consists of:
  - Catalogue with collections (e.g. Sentinel-2),
  - Collection has Items (e.g. Multiband image)
  - Items have assets (e.g. Single band image)



https://stacindex.org/catalogs/planet-labs-stac-catalog#/?t=1

Amazon Web Services (AWS)

- Common home for large geospatial datasets (e.g. Climate, Imagery, OpenStreetMap, Terrain)
- Typically data are hosted as "COGs" using "STAC"
- Access bands individually in seconds
- Can be accessed from a PC or from another cloud service
- There is a COST
- E.g.: https://aws.amazon.com/earth/



Google Earth Engine

- Available via: Python (geemap), R (rgee) and JavaScript
- Will use in next lab
- Very common in research and not-for-profit remote sensing
- Free for research education and not-for-profit use, must apply for commercial license


Remote Sensing of Environment 202 (2017) 18–27

Contents lists available at ScienceDirect

Remote Sensing of Environment

journal homepage: www.elsevier.com/locate/rse

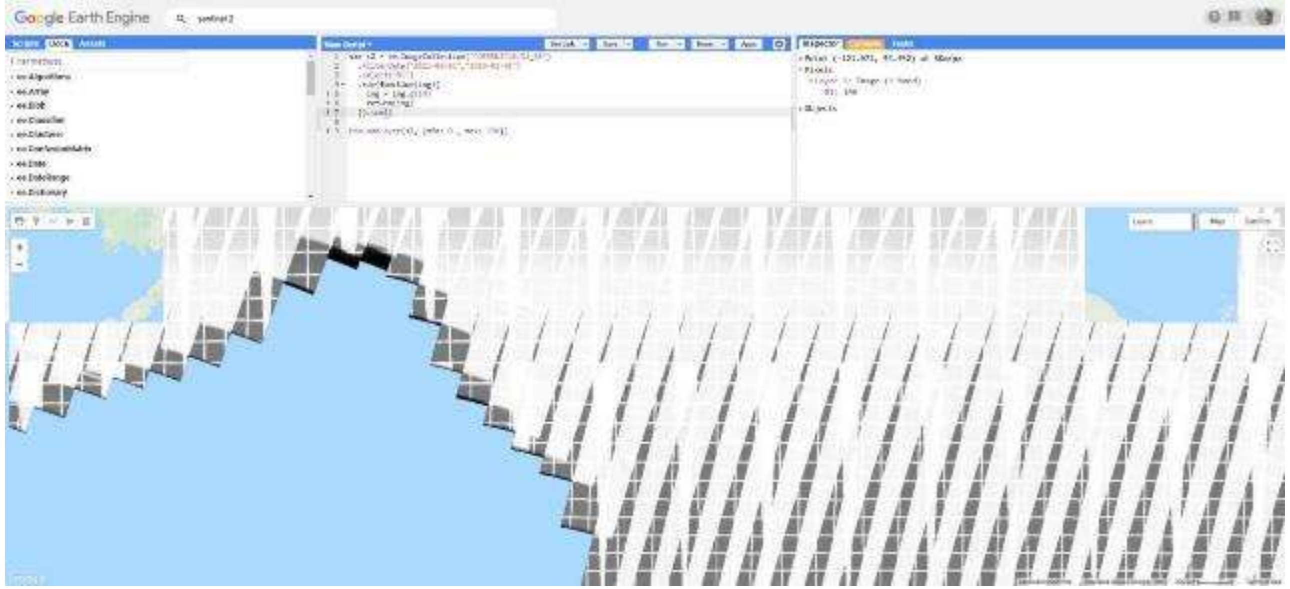
ELSEVIER

Google Earth Engine: Planetary-scale geospatial analysis for everyone

Noel Gorelick [a,*], Matt Hancher [b], Mike Dixon [b], Simon Ilyushchenko [b], David Thau [b], Rebecca Moore [b]

[a] Google Switzerland, Brandschenkestrasse 110, Zurich 8002, Switzerland
[b] Google Inc., 1600 Amphitheater Parkway, Mountain View, CA, 94043, USA

Cited >4k times!



High-Resolution Global Maps of 21st-Century Forest Cover Change

M. C. Hansen,[1*] P. V. Potapov,[1] R. Moore,[2] M. Hancher,[2] S. A. Turubanova,[1] A. Tyukavina,[1] D. Thau,[2] S. V. Stehman,[3] S. J. Goetz,[4] T. R. Loveland,[5] A. Kommareddy,[6] A. Egorov,[6] L. Chini,[1] C. O. Justice,[1] J. R. G. Townshend[1]

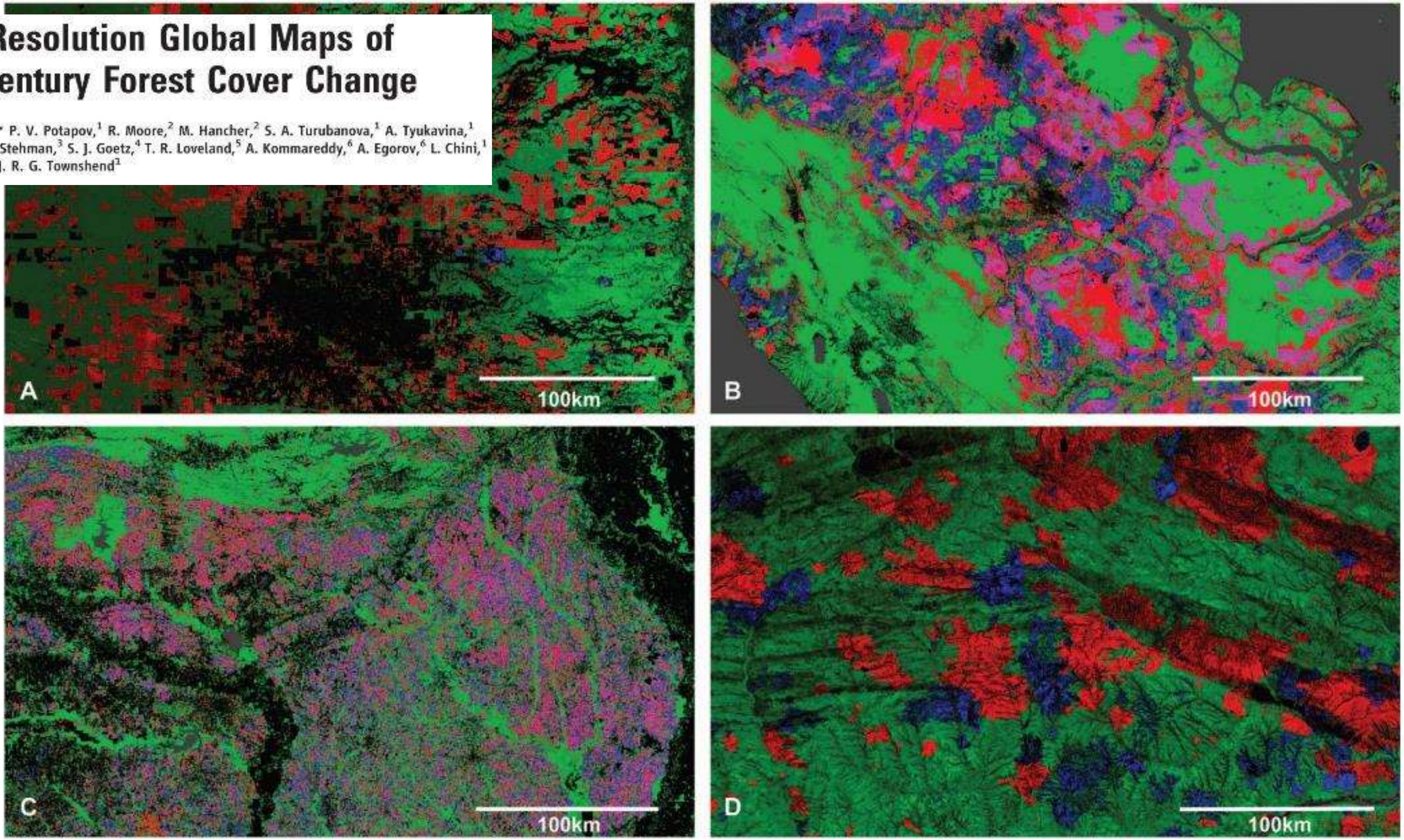
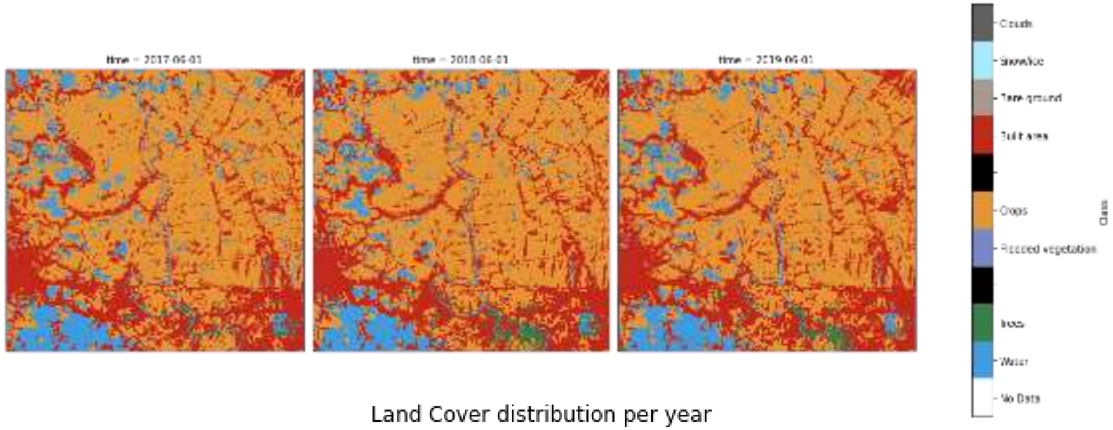


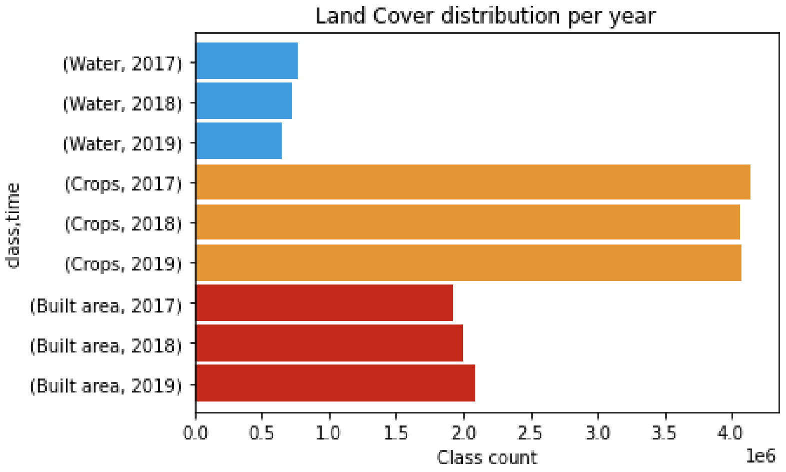
Fig. 2. Regional subsets of 2000 tree cover and 2000 to 2012 forest loss and gain. (**A**) Paraguay, centered at 21.9°S, 59.8°W; (**B**) Indonesia, centered at 0.4°S, 101.5°E; (**C**) the United States, centered at 33.8°N, 93.3°W; and (**D**) Russia, centered at 62.1°N, 123.4°E.

Cited >8k times!

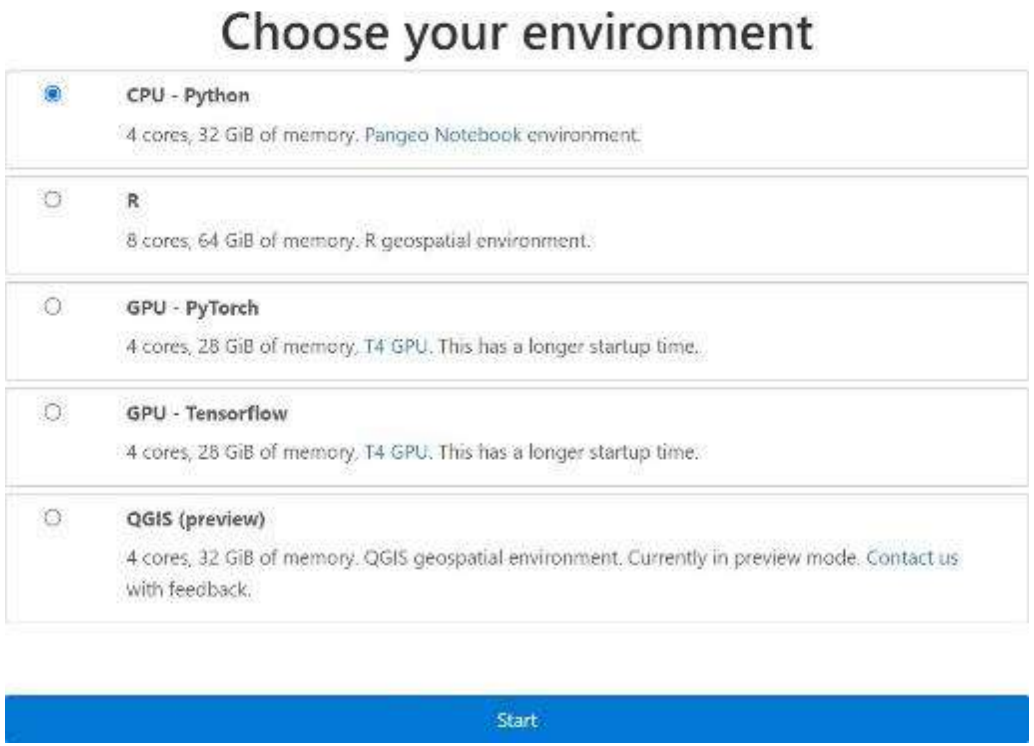
Microsoft Planetary Computer

The Planetary Computer combines a multi-petabyte catalog of global environmental data with intuitive APIs, a flexible scientific environment that allows users to answer global questions about that data, and applications that put those answers in the hands of conservation stakeholders.





https://planetarycomputer.microsoft.com/dataset/io-lulc-9-class#Example-Notebook



https://planetarycomputer.microsoft.com/

Conclusion

- Tricks exists to make local processing more efficient
  - More cores, more RAM, better GPU
  - Tiling large datasets
  - For visualization: 8-bit compression, VRTs, Overviews
  - Parallelization processing
- Cloud infrastructure keeps data close to processing
  - COGS/STAC to query catalogues and grab only the data you need
  - Some free/cheap options for cloud computing
  - Commercial solutions can be very efficient but can also be costly