



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')

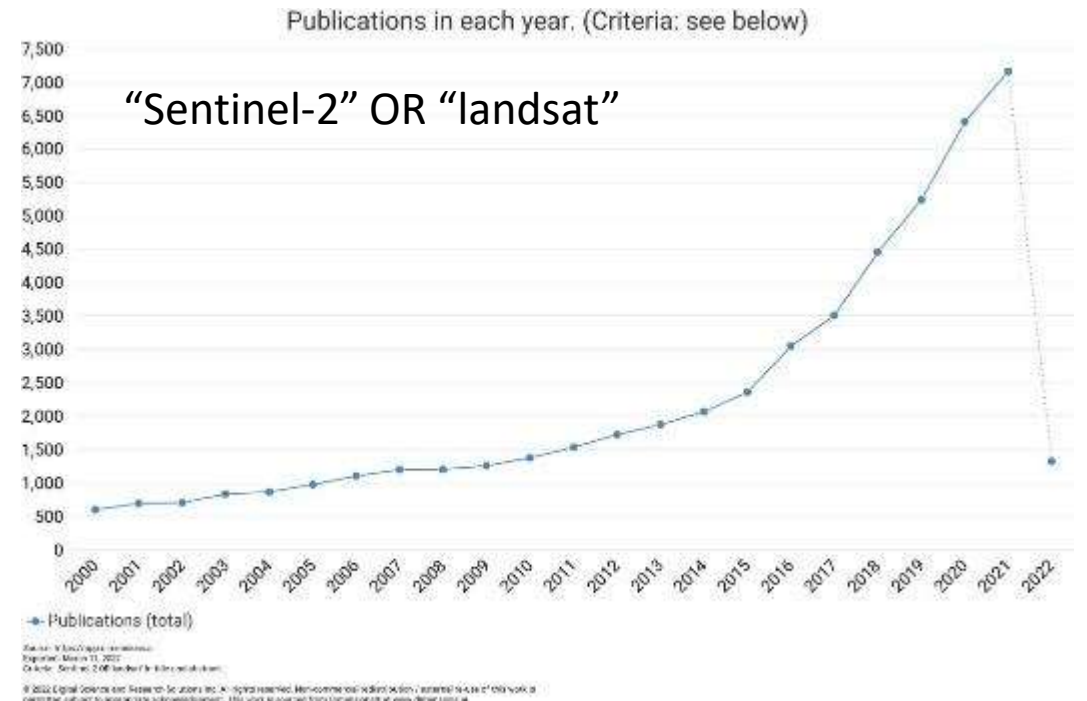
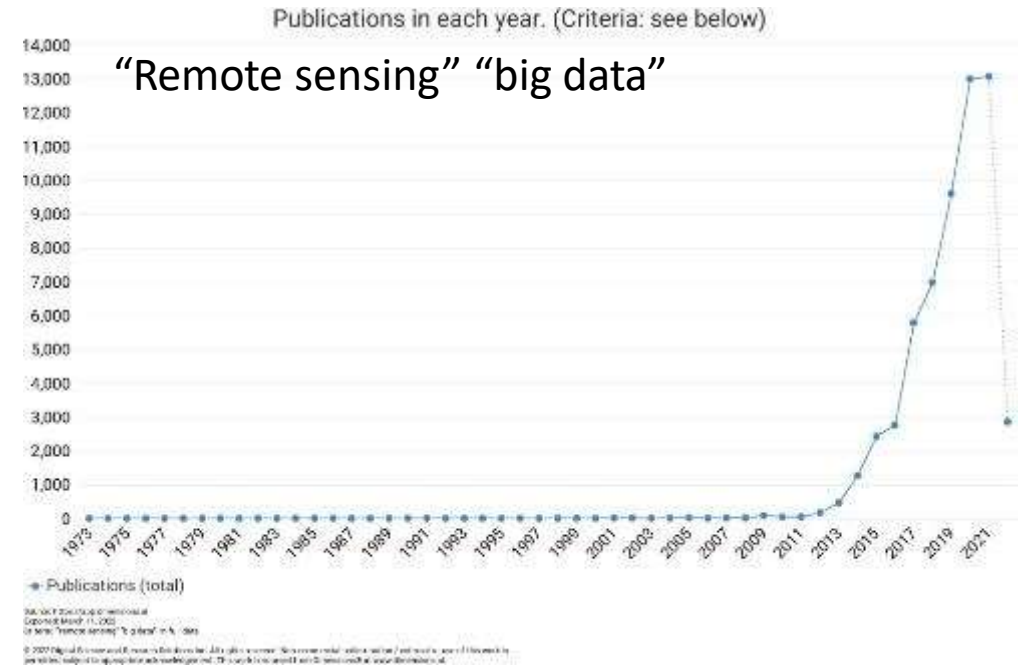
api.get_products_size(S1)

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

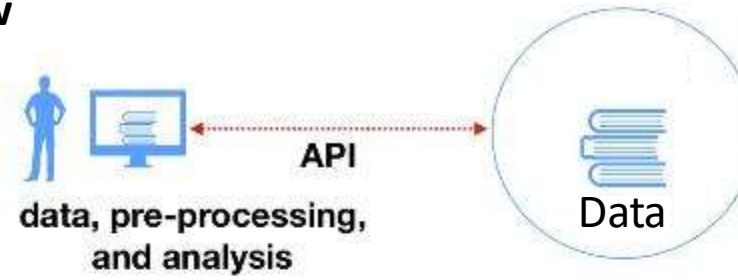
S2 = api.query(date=('2021-01-01T00:00:00Z', '2021-12-31T23:59:59Z'), platformname='Sentinel-2', producttype='S2MSI1C')

api.get_products_size(S2)

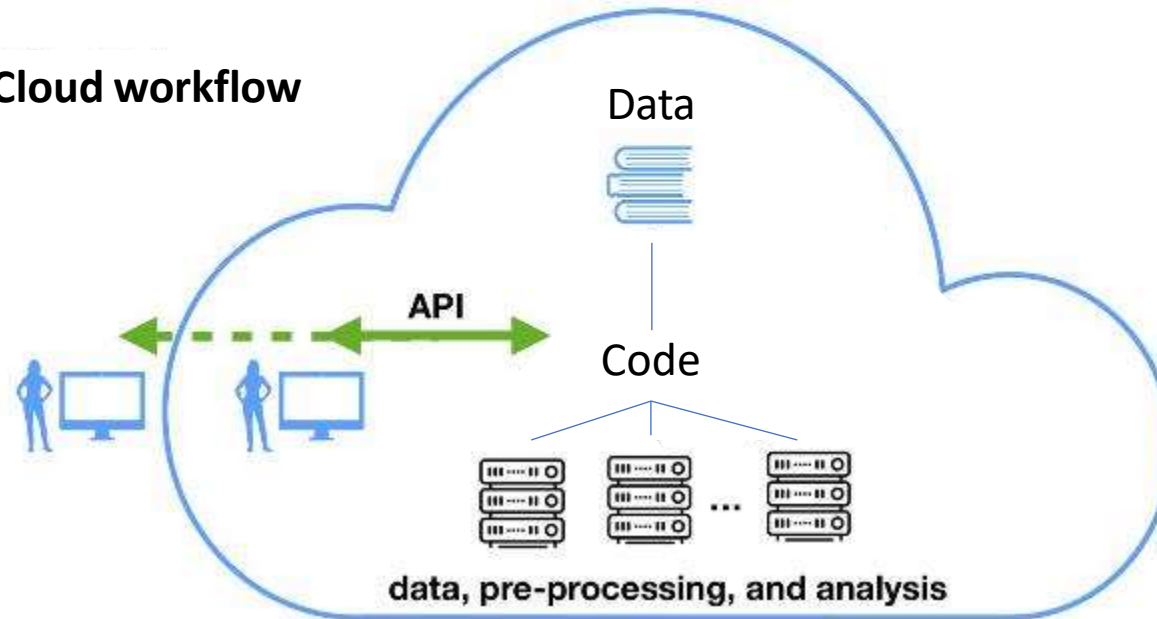
~ 0.94 Petabyte of Sentinel-2 L1C dat
```



Local workflow



Cloud workflow



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

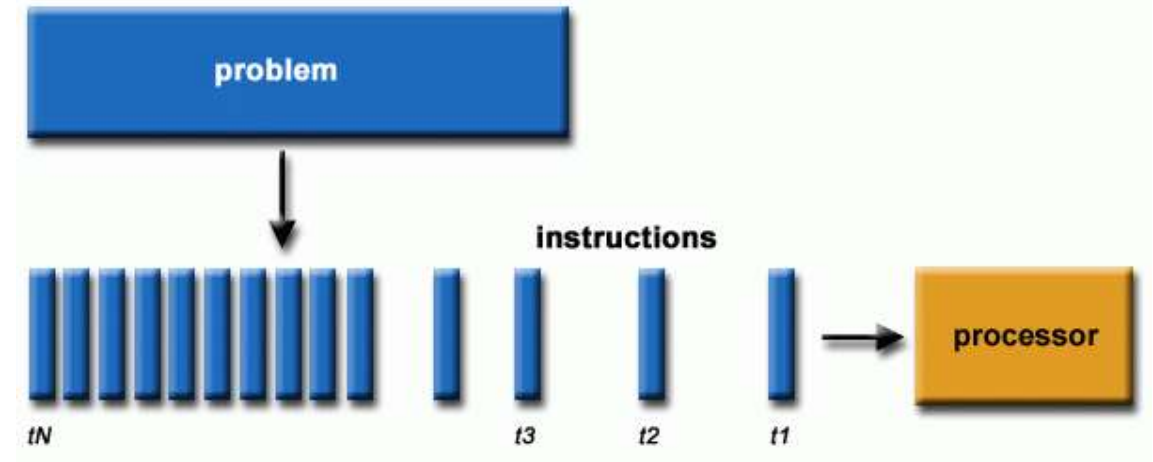
- **Parallel Computing**

- A problem is broken into parts
- Each part is broken into instructions
- Execute simultaneously on different processors
- Requires orchestration, sometimes not worth it

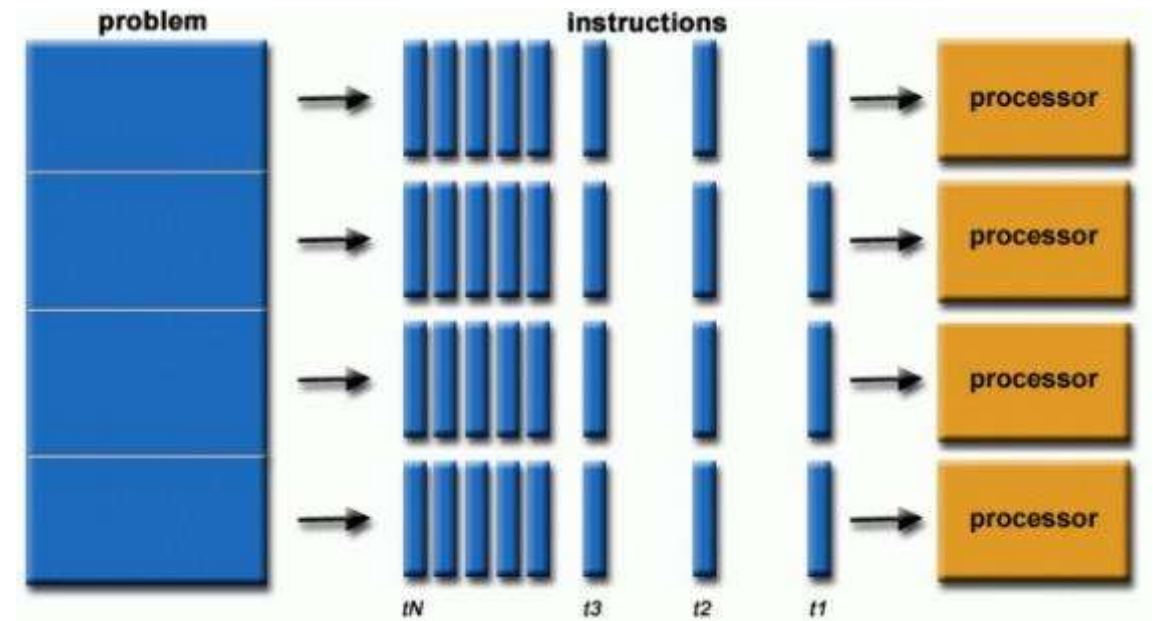
- **Hyper-threading**

- Better task scheduling
- Minimizes processor downtime
- Works for both serial and parallel computing
- Not equivalent to more cores

Serial computing

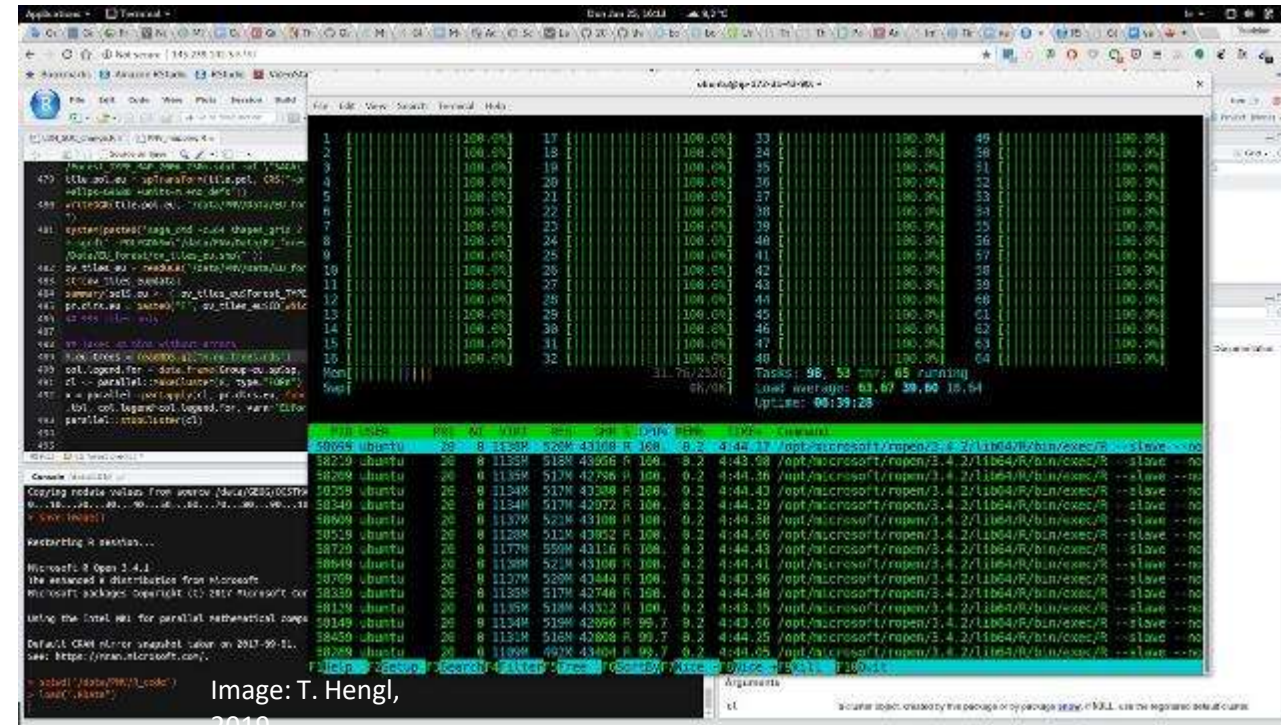


Parallel Computing



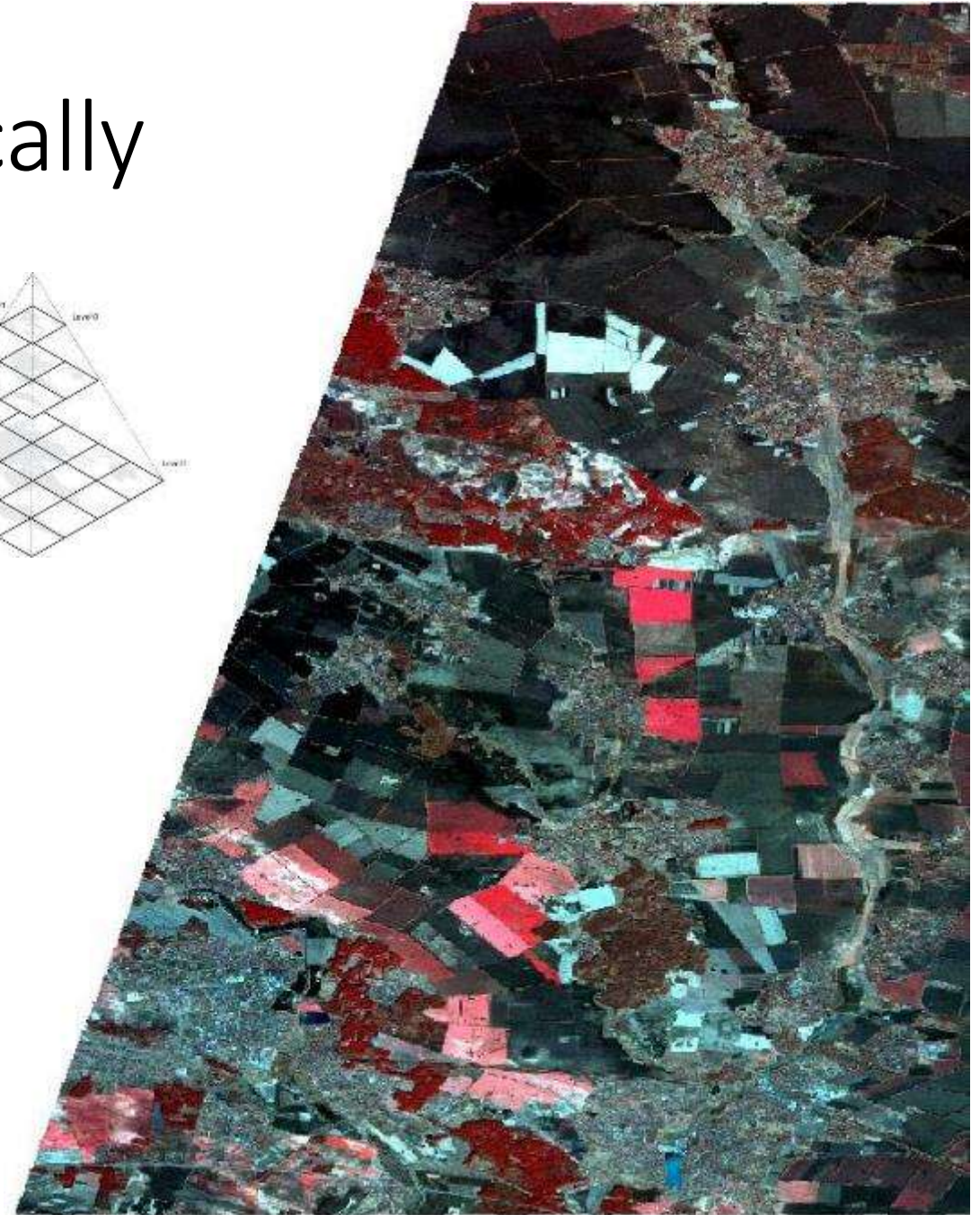
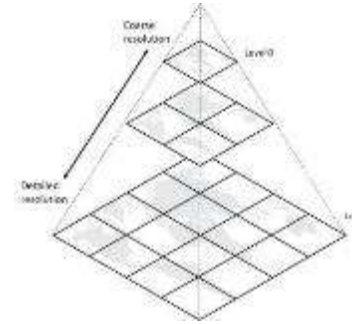
How to work in Parallel?

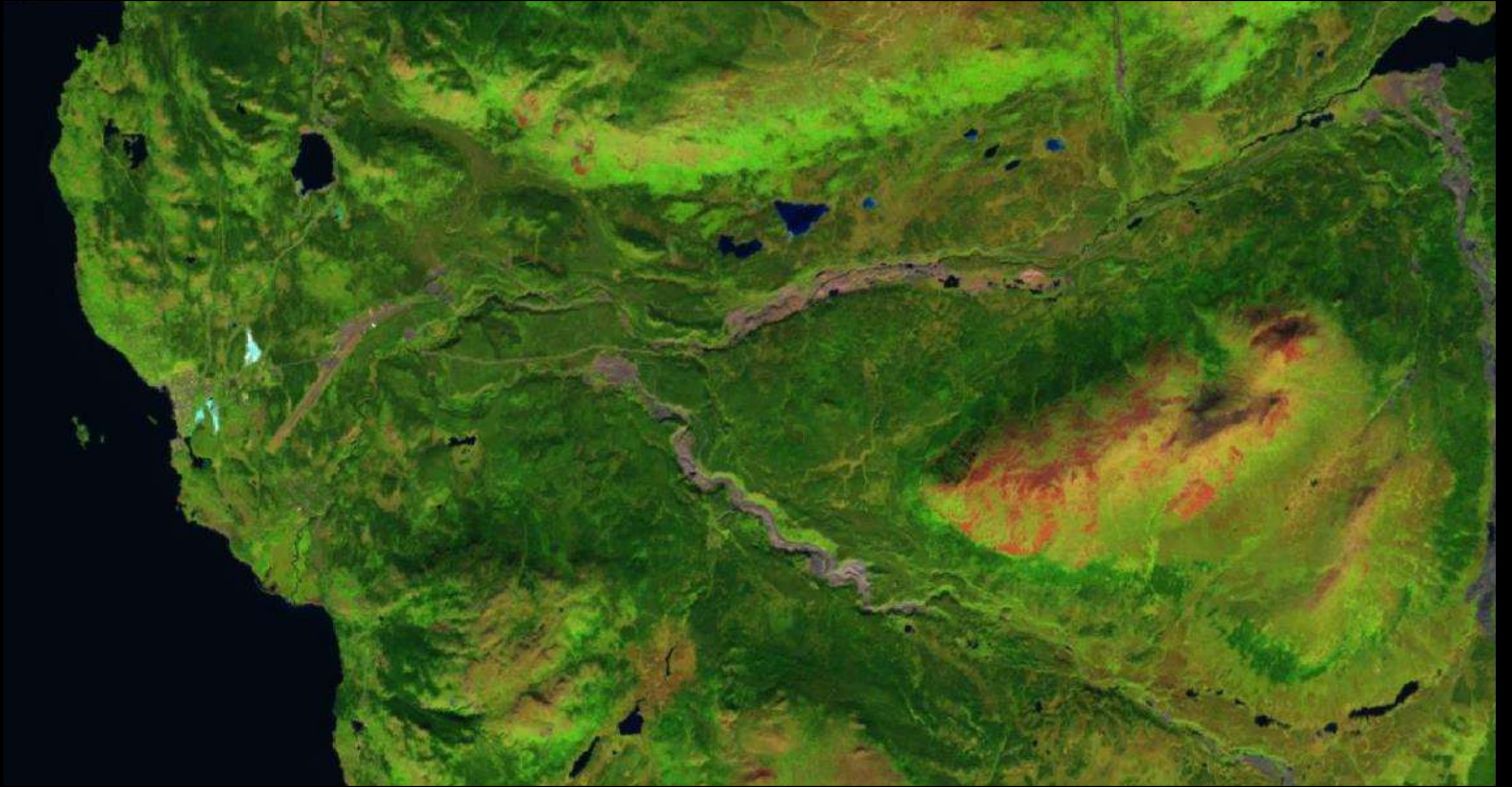
- Not all functions can be run in parallel
- SAGA GIS runs in parallel by default
- GRASS can be parallel with OpenMP
- Python: use `Dask`
- R: use `Future`
- ArcGIS: Available for some functions, not all



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

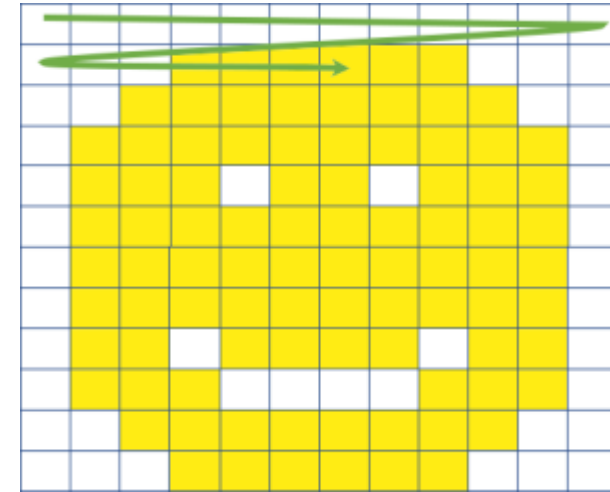




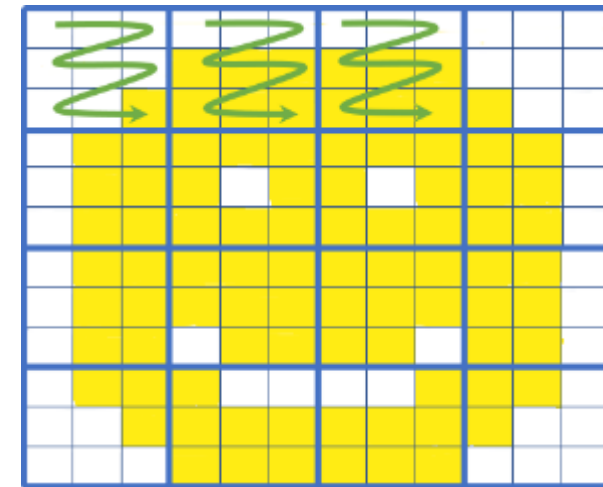
Cloud optimized geotiffs

- 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



44_cog_stac.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

For this demo, we will use data from <https://www.maxar.com/open-data/california-colorado-fires> for mapping California and Colorado. List of COGs can be found [here](#).

```
[ ] import ee
import geemap

[ ] Map = geemap.Map()
Map

Map(center=[40, -100], controls=(WidgetControl(options=['position'], widget=HBox(children=(ToggleButton(value=

[ ] url = 'https://opendata.digitalglobe.com/events/california-fire-2020/pre-event/2018-02-16/pine-gulch-fire20/10

[ ] geemap.cog_bounds(url)

[-108.63447456563128,
 38.963980238226654,
-108.38008268561431,
 40.025815049929754]

[ ] geemap.cog_center(url)

(-108.5072786256228, 39.49489764407821)

[ ] geemap.cog_bands(url)

['band1', 'band2', 'band3']

[ ] geemap.cog_tile(url)

'https://titiler.xyzcog/tiles/WebMercatorQuad/{z}/{x}/{y}@1x?url=https%3A%2F%2Fopendata.digitalglobe.com%2Fevent

[ ] Map.add_cog_layer(url, name="Fire (pre-event)")

[ ] url2 = 'https://opendata.digitalglobe.com/events/california-fire-2020/post-event/2020-08-14/pine-gulch-fire20/

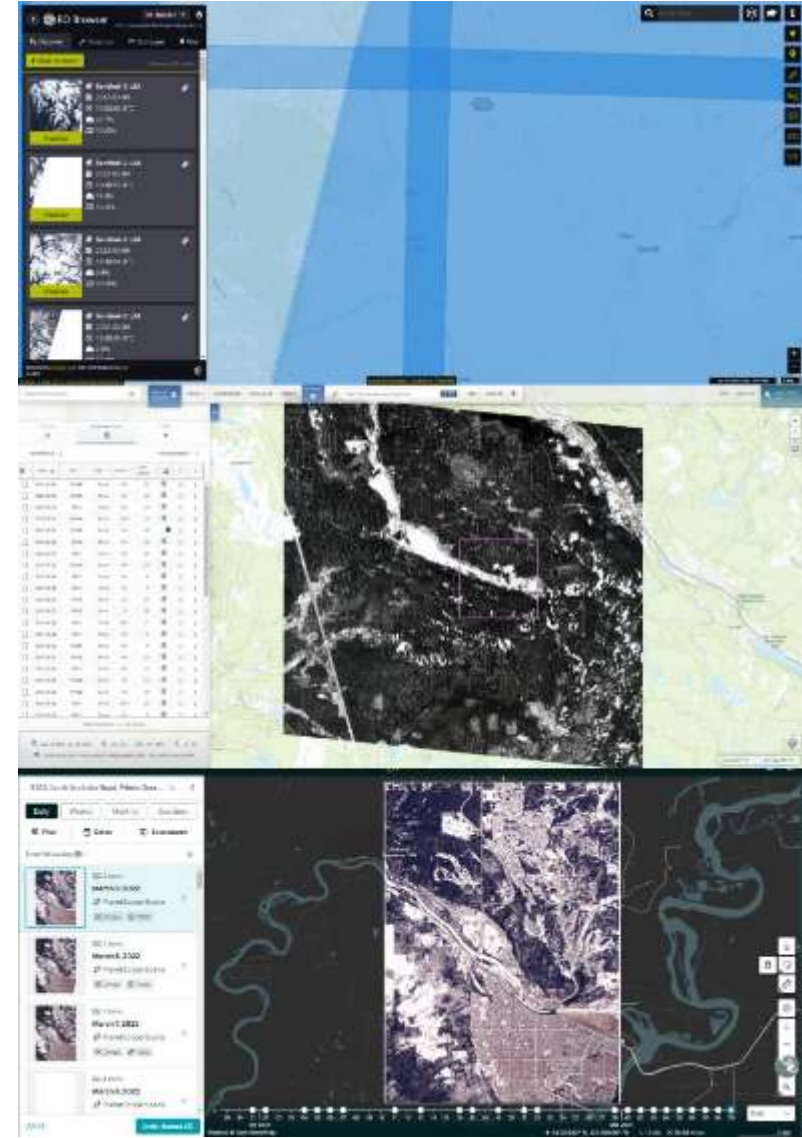
[ ] Map.add_cog_layer(url2, name="Fire (post-event)")
```

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/>

The screenshot shows the 'Registry of Open Data on AWS' page for Sentinel-2. The page has a dark header with the AWS logo and the title 'Registry of Open Data on AWS'. Below the header, there's a navigation bar with tabs: 'Applications', 'Dataset reports', 'Earth observation', 'Imagery', 'Natural resources', 'Satellite imagery', 'Data', and 'Earth science'. The 'Earth observation' tab is selected. The main content area is titled 'Sentinel-2' and contains several sections: 'Description' (explaining the Sentinel-2 mission and its data), 'Update Frequency' (stating that new data is added regularly), 'License' (indicating that access is free, full, and open), 'Documentation' (providing links to documentation), and 'Managed By' (listing the managing organization). To the right of the main content, there's a 'Resources on AWS' section with details about the data's location in AWS S3 buckets, including the Amazon Resource Name (ARN), AWS Region, and AWS CLI Access instructions. At the bottom, there's a 'Description' section for the data itself, including its format (GeoTIFF) and resolution (10m).

Registry of Open Data on AWS

aws

Sentinel-2

Applications Dataset reports Earth observation Imagery Natural resources Satellite imagery Data Earth science

Description

The Sentinel-2 mission is a land monitoring constellation of two satellites that provide high resolution optical imagery and provide continuity for the current SPOT and Landsat missions. The mission provides a global coverage of the Earth's land surface every 5 days, making the data of great use in on-going studies. L1C data are available from June 2015 onwards. L2A data are available from November 2015 over Europe, region and globally since January 2017.

Update Frequency

New Sentinel data are added regularly, usually within few hours after they are available on Copernicus OpenHub.

License

Access to Sentinel data is free, full and open for the broad Regional, National, European and International user community. View Terms and Conditions.

Documentation

Documentation is available for Sentinel-2 L1C and Sentinel-2 L2A.

Managed By

ESA

See all datasets managed by European Space Agency

Contact

<https://open.sentinel-hub.com/awsws/content>

How to Cite

Sentinel-2 was processed on date: from <https://registry.opendata.aws/sentinel-2>

License Examples

CC-BY-SA

Resources on AWS

Description

Level-1C scenes and metadata, in Requester Pays S3 bucket.

Resource type

S3 bucket

Amazon Resource Name (ARN)

`arn:aws:s3:::aws-registry-us-east-1/sentinel-2-l1c`

AWS Region

us-east-1

AWS CLI Access

`aws s3 cp --request-payer=requester s3://sentinel-2-l1c/`

Explore

Earth Search STAC Catalog

Description

S2 imagery files for L1C (SRC and CSF).

Resource type

S3 bucket

Amazon Resource Name (ARN)

`arn:aws:s3:::aws-registry-us-east-1/sentinel-2-l1c`

AWS Region

us-east-1

AWS CLI Access (No AWS account required)

`aws s3 cp --request-payer=requester s3://sentinel-2-l1c/sentinel-2-l1c/`

Description

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



Google Earth Engine: Planetary-scale geospatial analysis for everyone

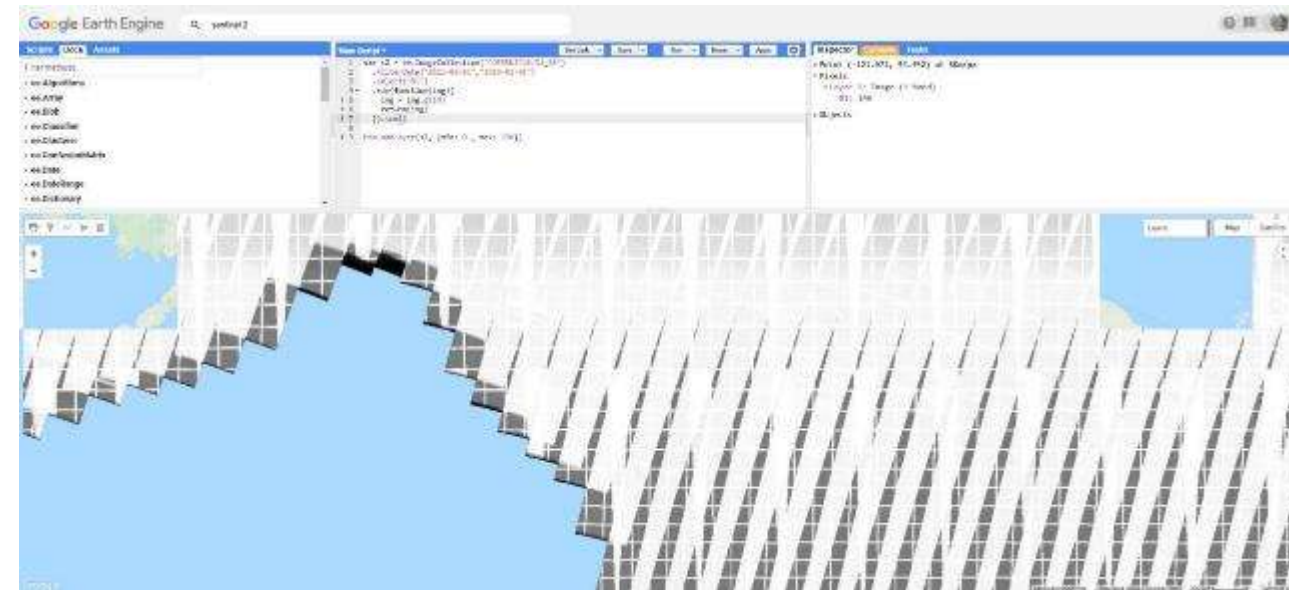
Noel Gorelick ^{a,*}, Matt Hancher ^b, Mike Dixon ^b, Simon Ilyushchenko ^b, David Thau ^b, Rebecca Moore ^b

^a Google Switzerland, Bundesstrasse 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. Townshend¹

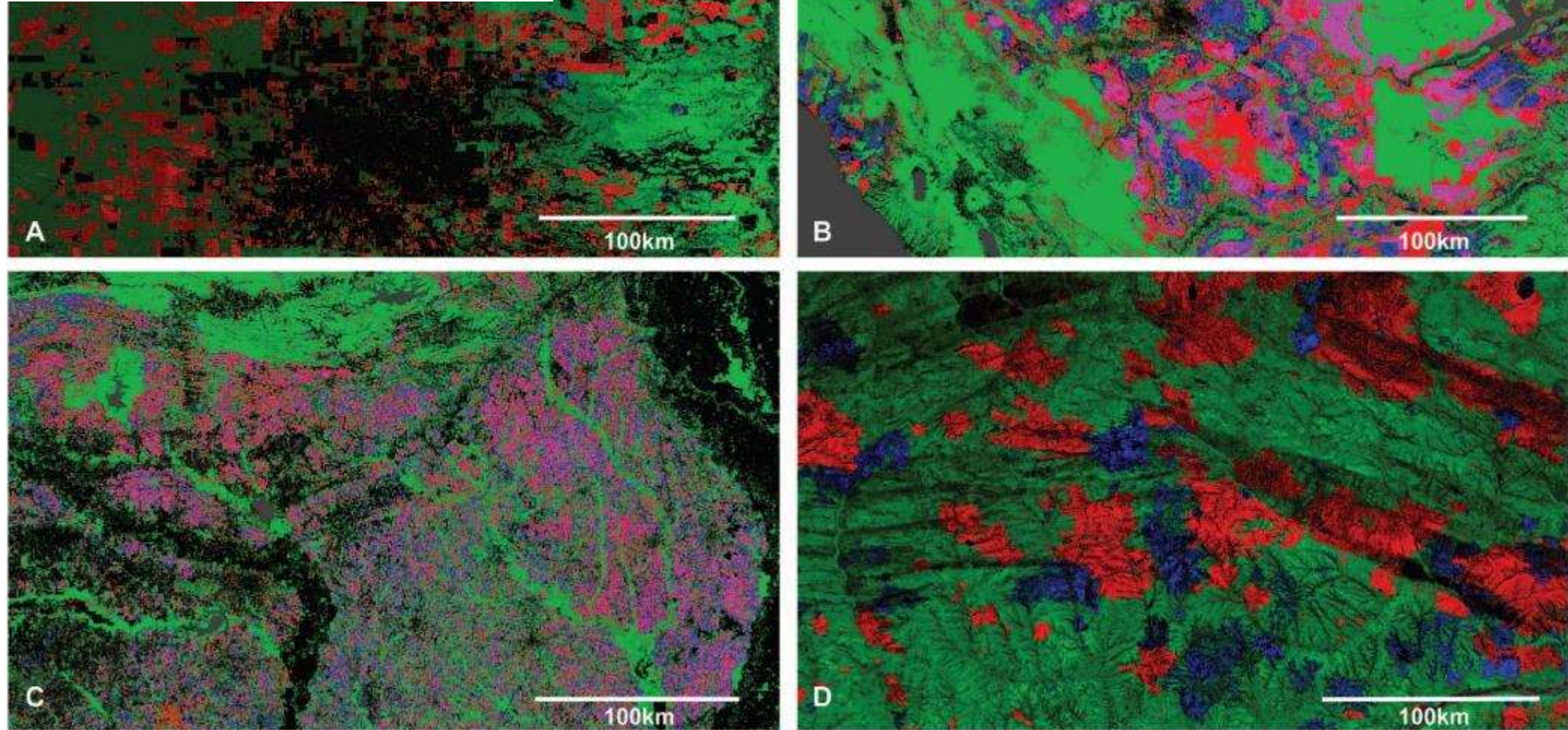
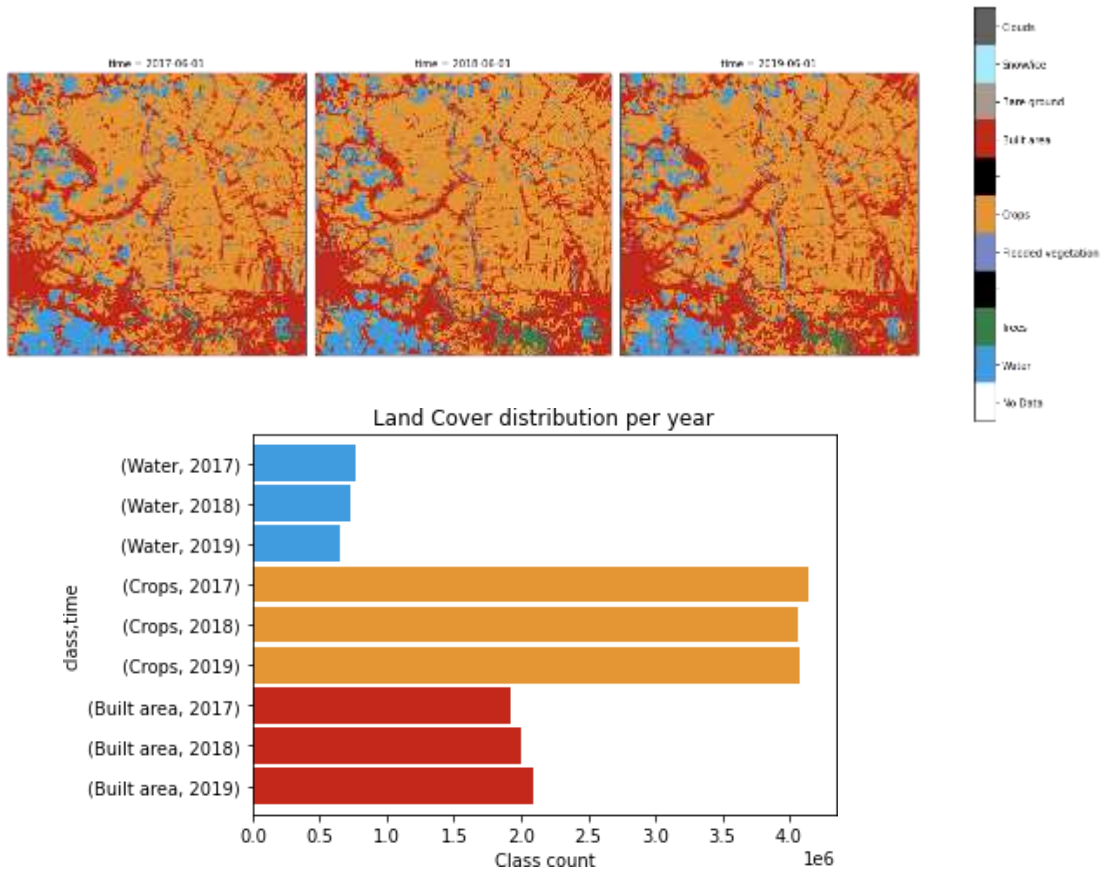


Fig. 2. Regional subsets of 2000 tree cover and 2000 to 2012 forest loss and gain. (A) Paraguay, centered at 21.9°S, 59.8°W; **(B)** Indonesia, centered at 0.4°S, 101.5°E; **(C)** the United States, centered at 33.8°N, 93.3°W; and **(D)** Russia, centered at 62.1°N, 123.4°E.

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.



Choose your environment

- ☒ **CPU - Python**
4 cores, 32 GiB of memory. Pangeo Notebook environment.
- ☐ **R**
8 cores, 64 GiB of memory. R geospatial environment.
- ☐ **GPU - PyTorch**
4 cores, 28 GiB of memory, T4 GPU. This has a longer startup time.
- ☐ **GPU - Tensorflow**
4 cores, 28 GiB of memory, T4 GPU. This has a longer startup time.
- ☐ **QGIS (preview)**
4 cores, 32 GiB of memory. QGIS geospatial environment. Currently in preview mode. Contact us with feedback.

[Start](#)

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