Analyzing massive amounts of EO data in the cloud with R, gdalcubes, and STAC

Marius Appel

OpenGeoHub Summer School 2021
Sept. 1, 2021

Institute for Geoinformatics
University of Münster

Motivation

- Data availability (e.g. Sentinel-2) in the cloud
- Method availability (e.g. in R, > 18k CRAN packages)
- No time (frustration tolerance?) for downloading > 100 GB from data portals

How to avoid data downloads and make use of the data and method availability ?

Tutorial overview

Objective: Show **how** you can analyze satellite image collections in the cloud with R

1. **Introduction**: Cloud computing, EO data in the cloud, STAC, COG, and gdalcubes
2. **Live examples**
    1. Creating composite images
    2. Complex time series analysis
3. **Discussion**

All **materials** are available on GitHub: https://github.com/appelmar/ogh2021.

"... in the cloud"

Services:

- Google Earth Engine (GEE)
- Sentinel Hub
- openEO backends
- ...

Infrastructure providers:

- Amazon web services (AWS)
- Google Cloud Platform
- Microsoft Azure
- Copernicus DIASes
- ...

In this tutorial, we will use a custom machine on AWS to analyze EO data *in the cloud*.

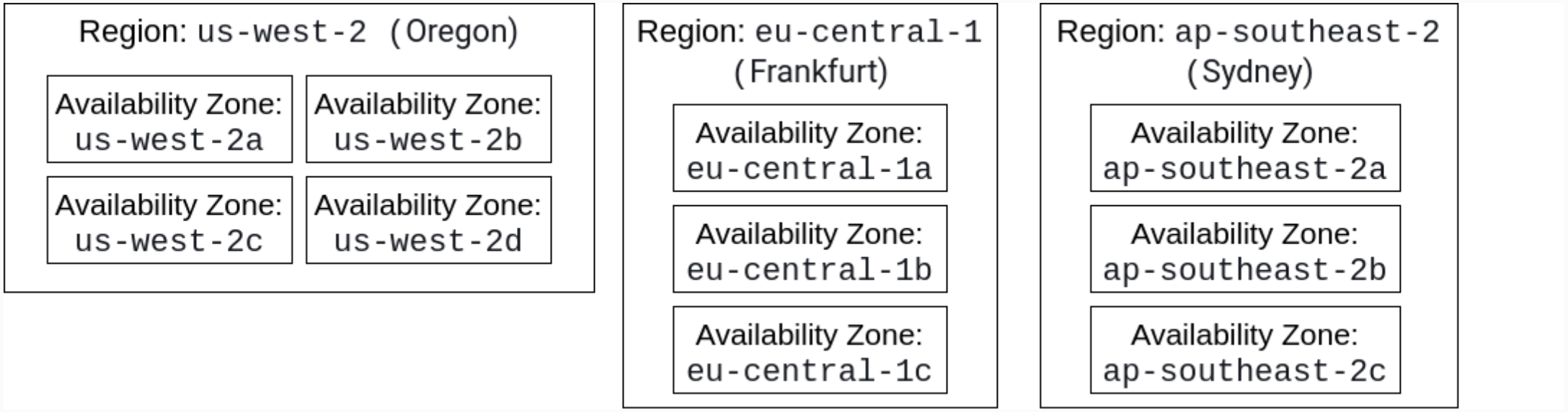
Open EO data on cloud-computing platforms (examples)

Provider	Data
Amazon web services (AWS)	Sentinel, Landsat, ERA 5, OSM, CMIP 6, and more, see here
Google Cloud Platform	Landsat, Sentinel, access to GEE data
Copernicus DIASes	Sentinel + more (see here)

Getting started with Amazon web services (AWS)

AWS overview

- Lots of separate data centers with large clusters



- In total: 25 regions and 81 availability zones
- Basic service to run (virtual) machines: EC2 (Amazon Elastic Compute Cloud)

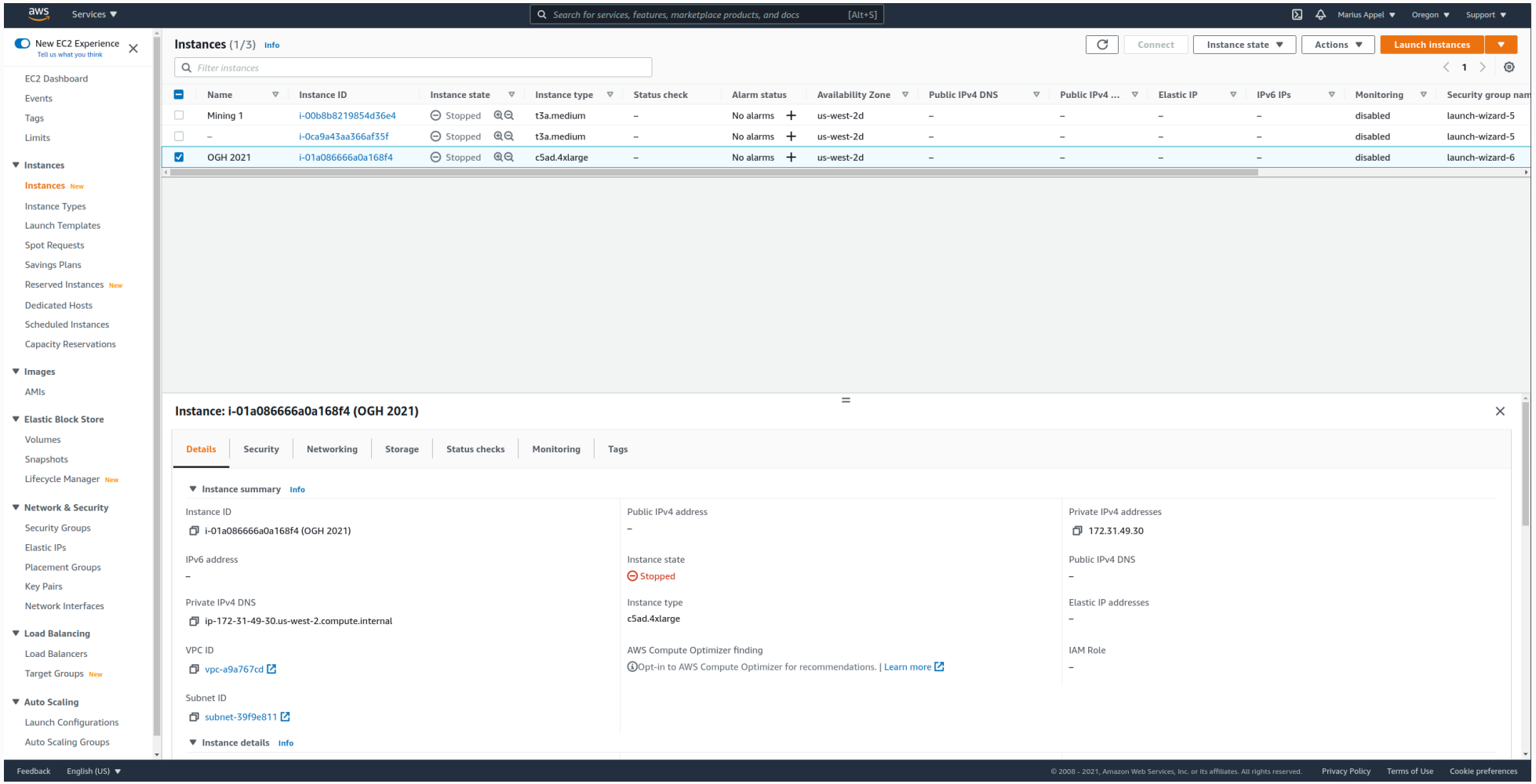
Basic steps to run a machine on AWS

1. Select a region and machine instance type, based on costs, hardware, and OS
2. Create a key pair for accessing the machine over SSH
3. Click "Launch instance" and follow instructions
4. Connect via SSH and install software (PROJ, GDAL, R, RStudioServer[1], R packages, ...)

*Notice that security considerations (e.g. by using IAM roles, multi-factor authorization) are **NOT** part of this tutorial.*

[1] You need to add a security rule to allow public / protected access to RStudioServer.

AWS Management Console



EO data in the cloud

Object Storage: S3

EC2 machines have local storage (EBS) but big data archives use highly scalable **object storage**.

S3 elements:

- **Bucket**: container for objects that is stored in a specific AWS region
- **Objects**: Individual files and corresponding metadata within a bucket, identified by a unique key
- **Key**: Filenames / Path or similar; unique within a bucket

Pricing (storage, transfer, requests):

- Bucket owner pays by default
- For **requester pays** buckets, transfer and requests are paid by users

Examples

Buckets:

- https://registry.opendata.aws/sentinel-2
- https://registry.opendata.aws/usgs-landsat/

Object:

- http://landsat-pds.s3.amazonaws.com/L8/003/017/LC80030172015001LGN00/LC80030172015001LGN00_B9.TIF

Accessing imagery

- Buckets are **not** a drive on your machine
- Data access over HTTP requests (PUT, GET, DELETE, ...)

Challenges

1. How to find images by location, time, and other criteria?

2. How to efficiently read image data from S3 without copying images to our machine storage first?

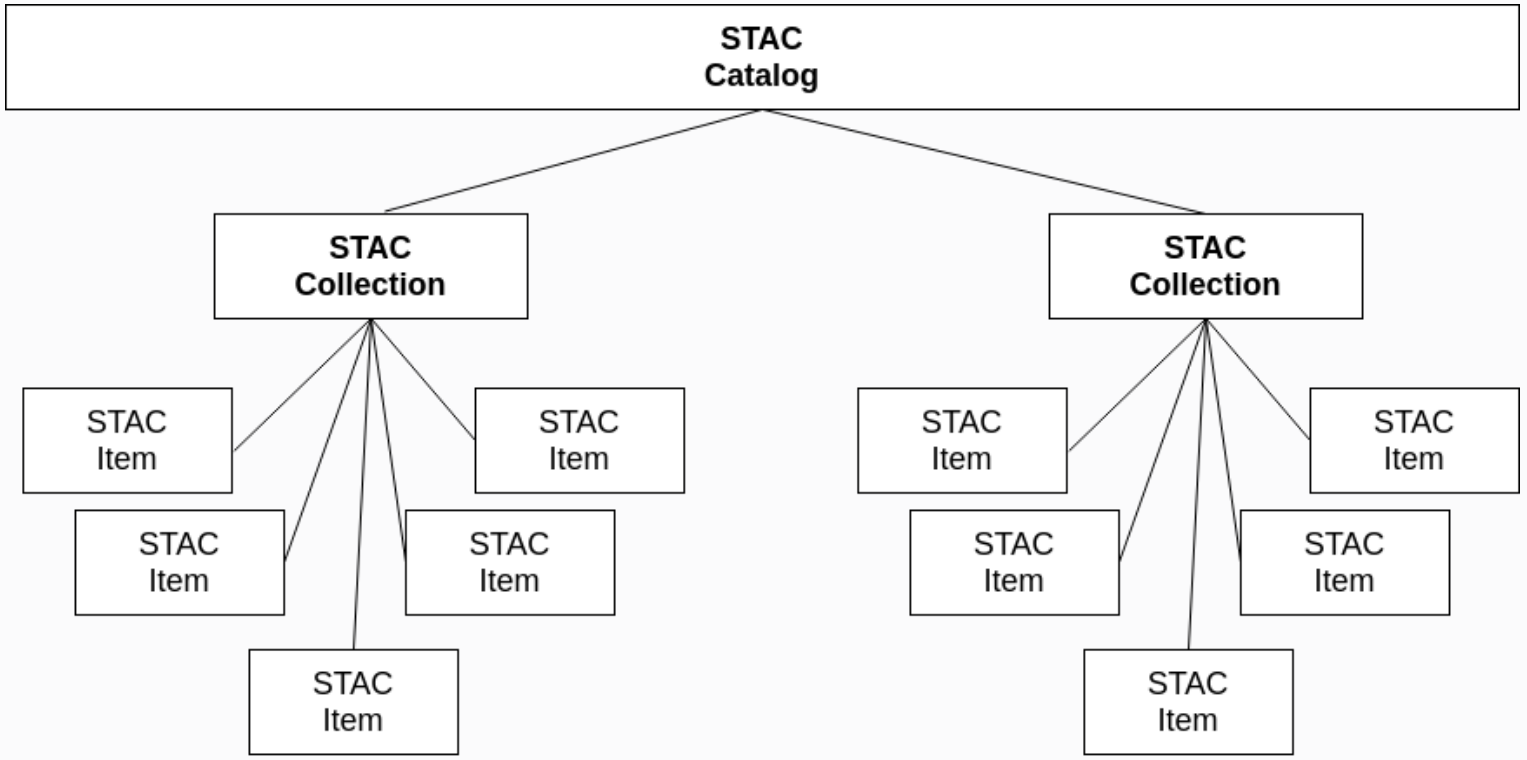
Cloud-native satellite imagery access with STAC-API and COGs

STAC

- standardized JSON-based language for describing catalogs of **spatiotemporal** data (imagery, point clouds, SAR)
- extensible (available extensions include EO, Data Cubes, Point Clouds, and more)
- 1.0.0 release available since May 2021
- growing ecosystem

SpatioTemporal
Asset Catalog

STAC



- **Items** are inseparable objects of data (assets) and metadata (e.g. a single satellite image)
- **Catalogs** can be nested
- **Collections** extend catalogs and can be used to group items and their metadata (e.g. license)

STAC

Static STAC catalogs

- Typically set of linked JSON files, starting with a `catalog.json`
- Catalog JSON contains links to collections, nested catalogs, or items
- Items contain assets (links to files) and metadata
- Problem: All items must be processed for searching
- Example: https://landsat.stac.cloud/?t=catalogs*

STAC API

- Web-service for dynamic search of STAC items by area of interest, datetime, and other metadata
- Compliant with OGC API - Features standard

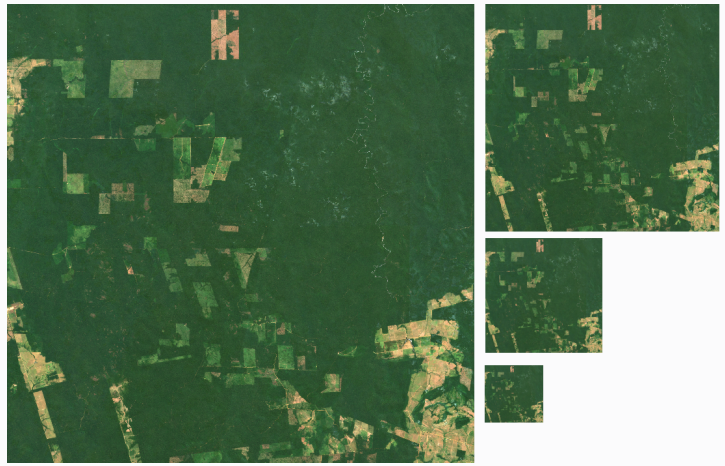
STAC Index

- A good starting point to find available STAC collections and API services: https://stacindex.org

Cloud-optimized GeoTIFF (COG)

Image file formats must be cloud-friendly to reduce transfer times and costs associated with transfer and requests

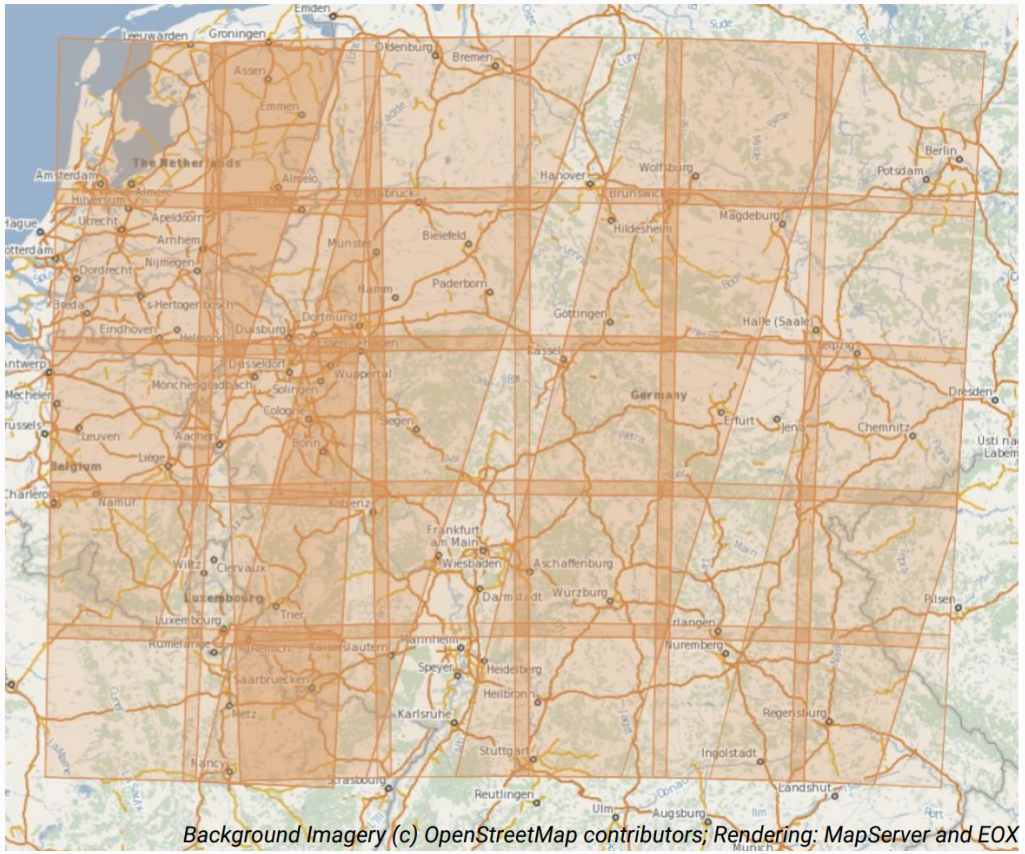
- COG = Normal **tiled GeoTIFF** files whose content follows a **specific order of data and metadata** (see full spec here)
- support compression
- support efficient **HTTP range requests**, i.e. partial reading of images (blocks, and overviews) over cloud storage
- may contain overview images (image pyramids)



- GDAL can efficiently read and write COGs, and access object storage in the cloud with *virtual file systems*

On-demand data cubes with the gdalcubes library

Satellite image collections

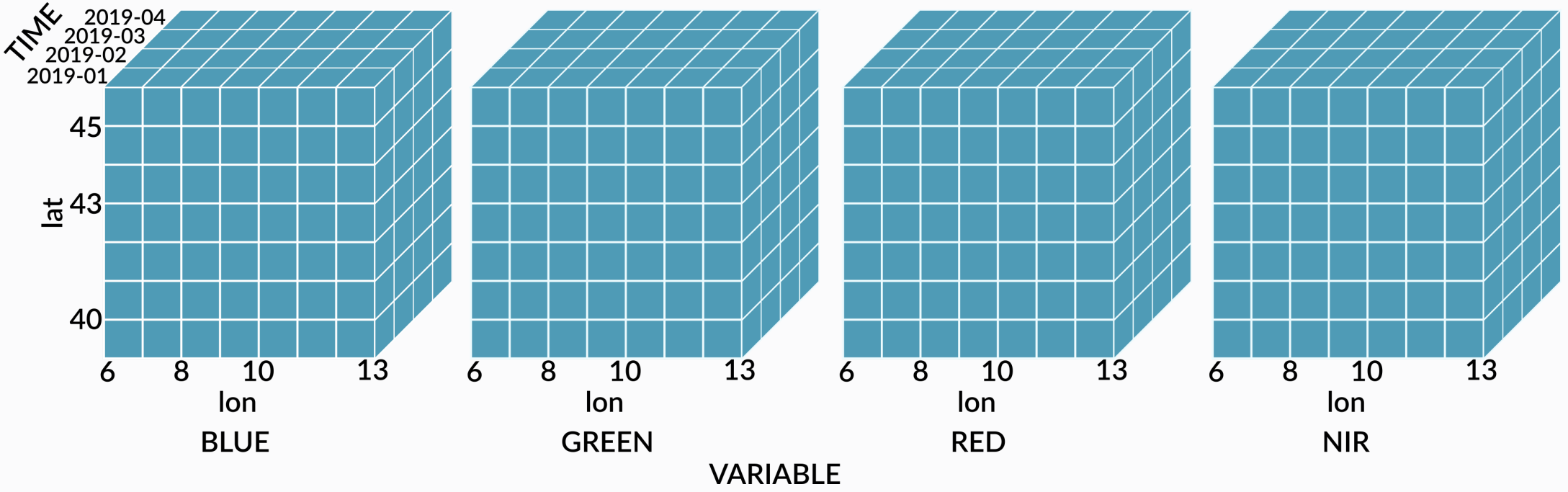


Background Imagery (c) OpenStreetMap contributors; Rendering: MapServer and EOX

Images spatially overlap, have different coordinate reference systems, have different pixel size depending on spectral bands, yield irregular time series for larger areas

What is a data cube?

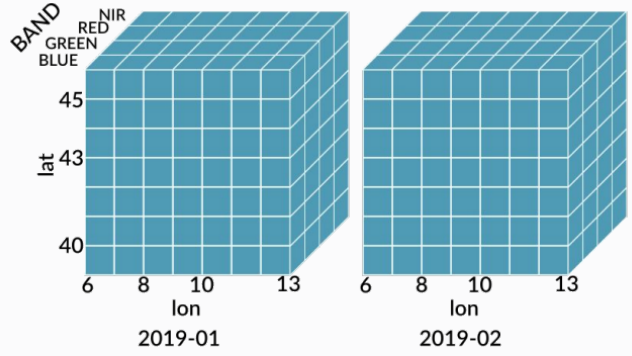
Here: **A four-dimensional (space, time, variable / band) regular raster data cube**

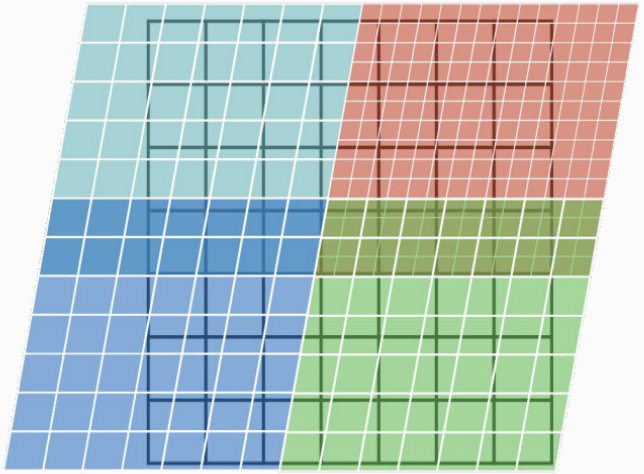


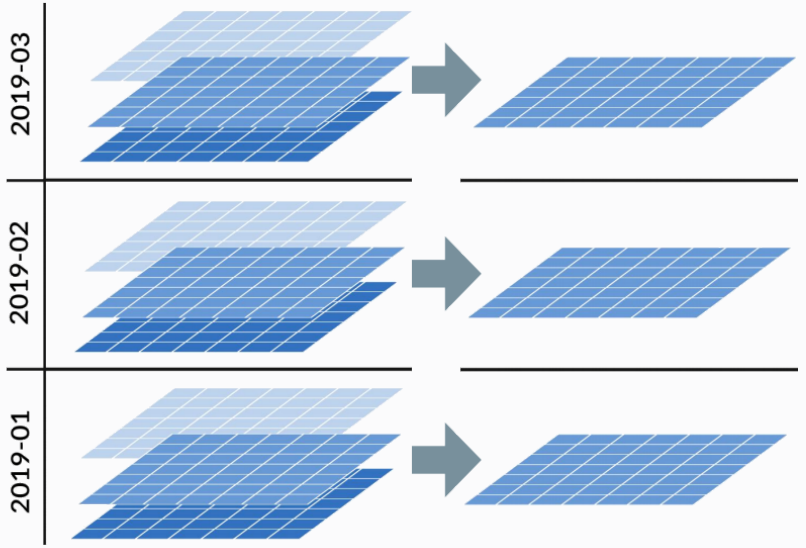
- collect all observations in one object
- $b \times t \times y \times x \rightarrow$ real value
- single CRS, cells have constant temporal duration, and spatial size

Data Cube creation is lossy

Define Data Cube Parameters → **Image Warping** → **Temporal Aggregation**







There is no single correct data cube!

gdalcubes

- Open source C++ library and R package available from CRAN
- Creation and processing of data cubes from satellite image collections[1]
- Image data is read on the fly, when needed (lazy evaluation), multithreaded, chunk-wise
- Uses GDAL to read and warp data
- Additional features: user-defined R functions on pixels / pixel time series, pixel masking combining data from different image collections

[1] Appel, M., & Pebesma, E. (2019). On-demand processing of data cubes from satellite image collections with the gdalcubes library. Data, 4(3), 92.

Live examples

see https://appelmar.github.io/ogh2021/tutorial.html

Discussion

Pros and cons of EO data analysis in the cloud

Workflow is based on the existence of STAC-API services and imagery as COGs!

Advantages

- Access to huge data archives
- Flexibility: You can do whatever you can do on your local machine
- Powerful machines available
- Open source software only

Disadvantages

- Not free
- GEE and others are easier to use (some are free)
- Your institution's computing center might have more computing resources (*for free*)
- Setup and familiarization needed

Summary

- Cloud-computing platforms contain lots of open EO data
- Cloud storage differs from local storage
- Technology and tools:
  - STAC (and STAC API!) for efficient and standardized search of spatiotemporal EO data
  - COGs allow efficiently reading parts of imagery, potentially on lower resolution
  - GDAL has everything for efficient data access on cloud storage
  - gdalcubes makes the creation and processing of data cubes from satellite image collections in R easier

Thanks!

Slides and Notebook:

https://github.com/appelmar/ogh2021

Contact:

@appelmar
marius.appel@uni-muenster.de

Slides created via the R packages: xaringan, *gadenbuie/xaringanthemer*