Analyzing Multi-Variable Earth Observation Data Cubes

Geospatial Sensing 2020

Marius Appel

Aug 31, 2020

WWU MÜNSTER

ifgi
Institut für Geoinformatik
Universität Münster

Background

- Research associate and PhD student in Geoinformatics at University of Münster, Germany
- PhD topic "*Facilitating Statistical Analysis of Large Satellite-Based Earth Observation Data*"

- Main Interests:
    - Spatial Statistics / Spatial Data Science
    - Earth observation analytics (data cubes, array databases)
    - Applications in environmental modeling
    - Web development

- Open source software development (mostly in C++ and R)

Aim of this tutorial

- Overview of Earth Observation (EO) data cubes
- Demonstration of the gdalcubes library
- Present applications of data cubes for combining data from different EO missions
- Discuss challenges and ongoing developments

Contents

Introduction

Earth Observation Data Cubes

Motivation

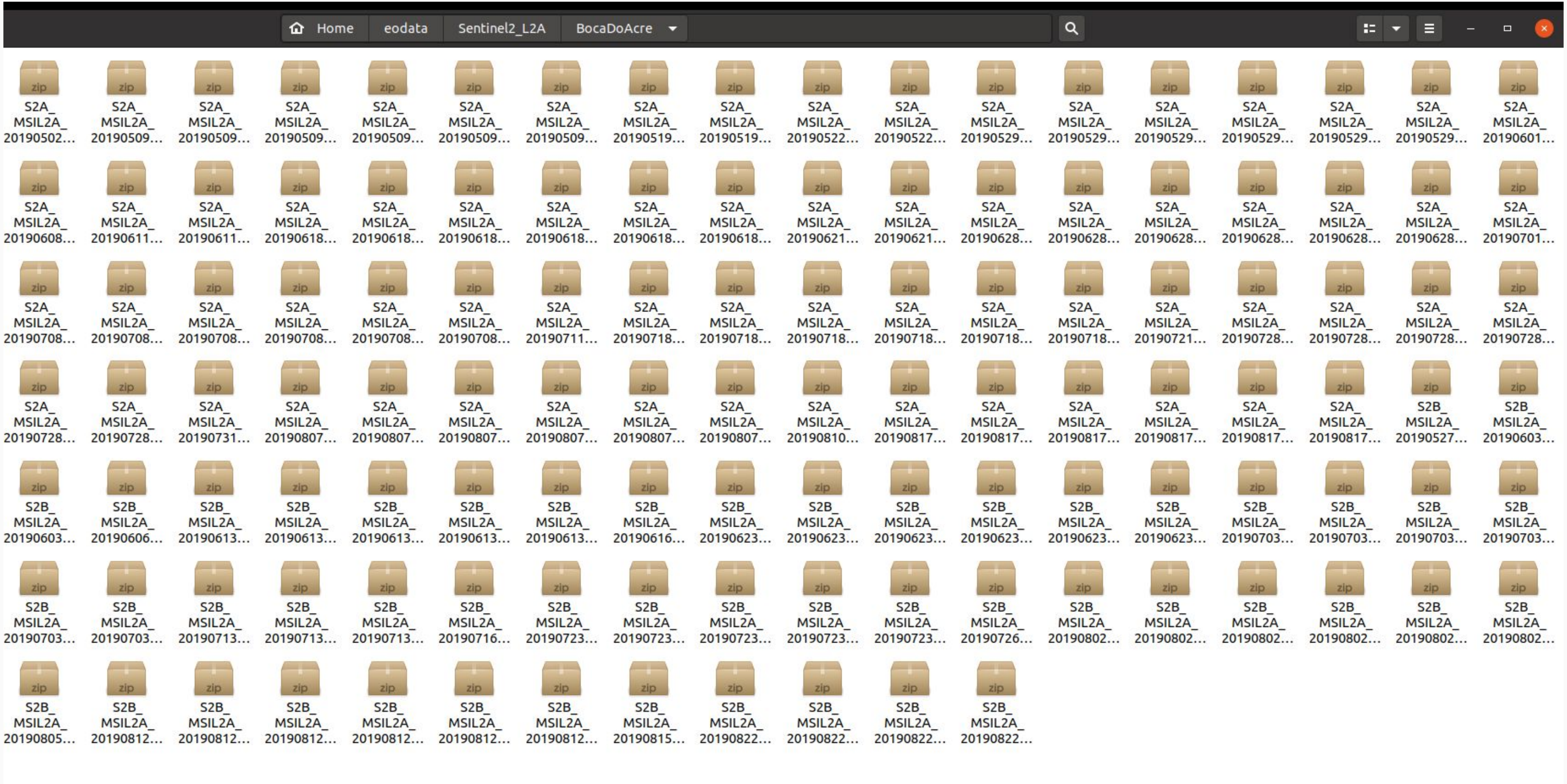
Availability of Earth Observation Data

Availability of Analysis Methods (ML, Statistics)

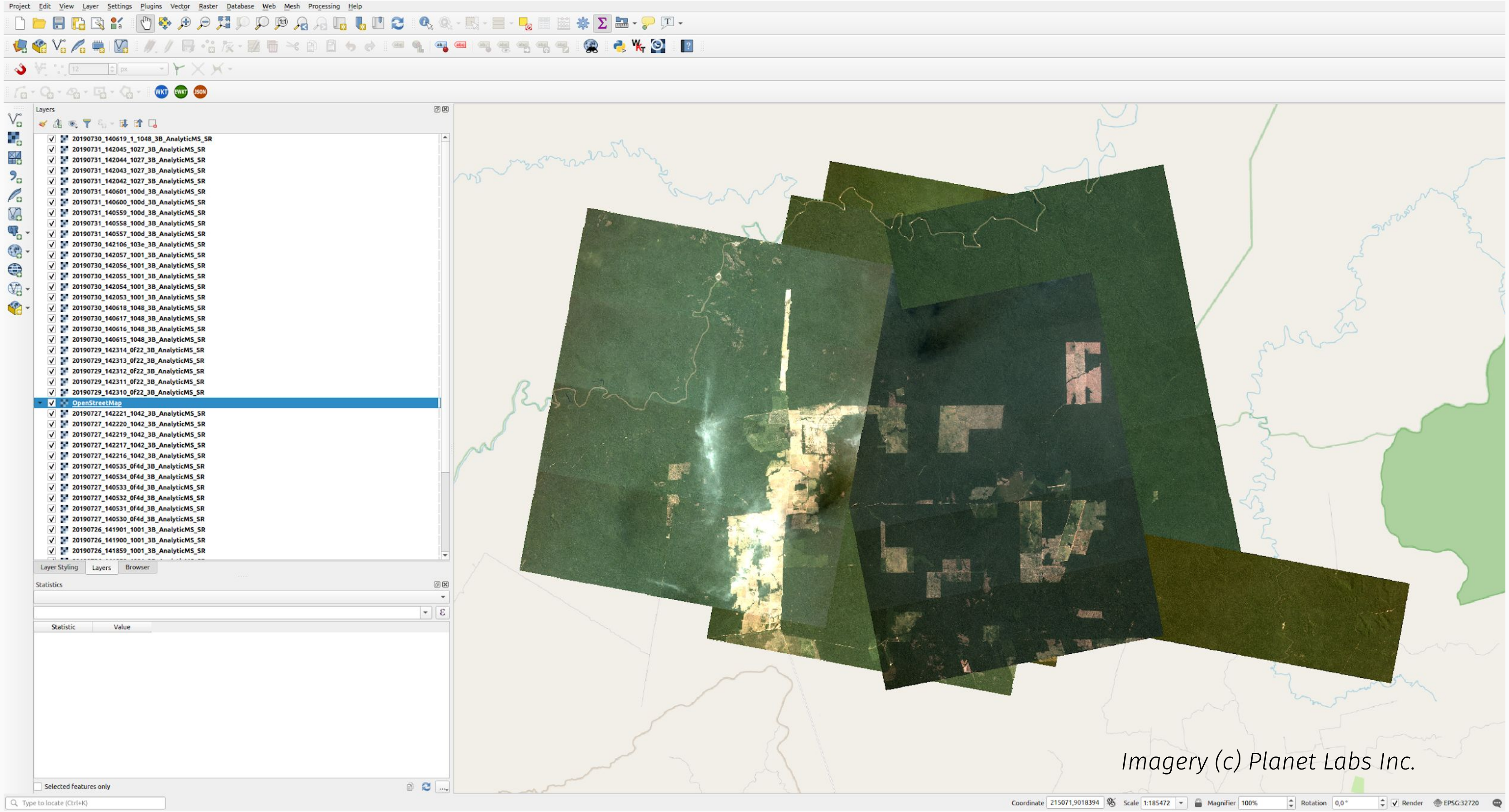
Researcher / Data Scientist / ...

How to apply complex analysis methods on today's (satellite-based) Earth observation datasets?

Analysis ready?



Analysis ready?

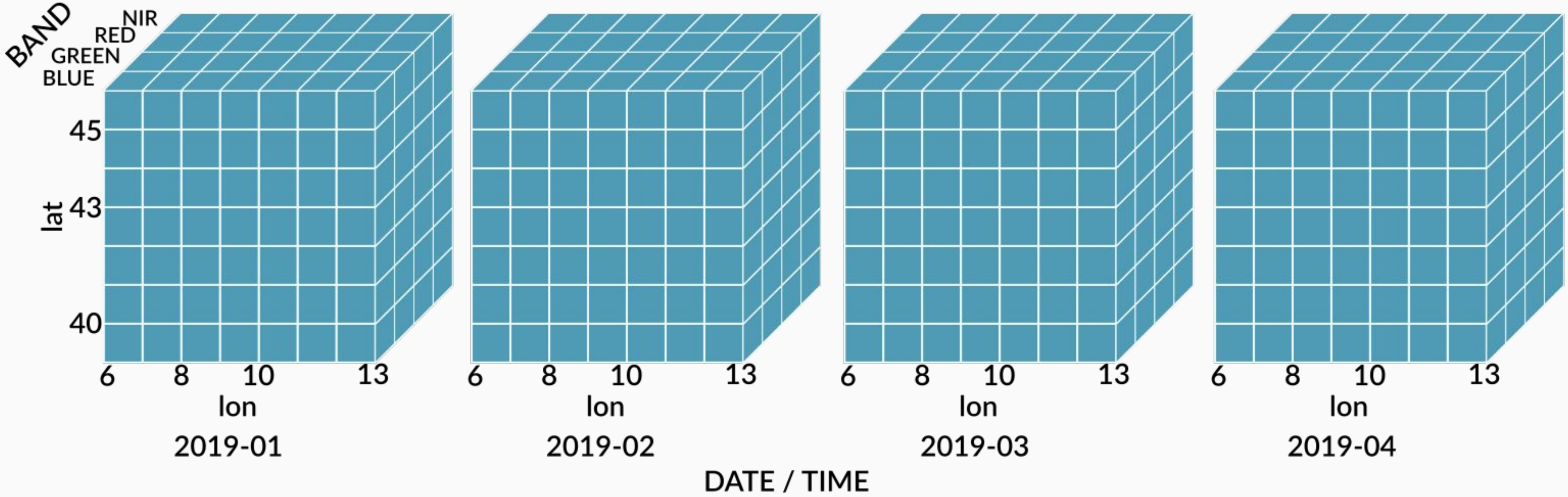


Imagery (c) Planet Labs Inc.

Analysis ready?

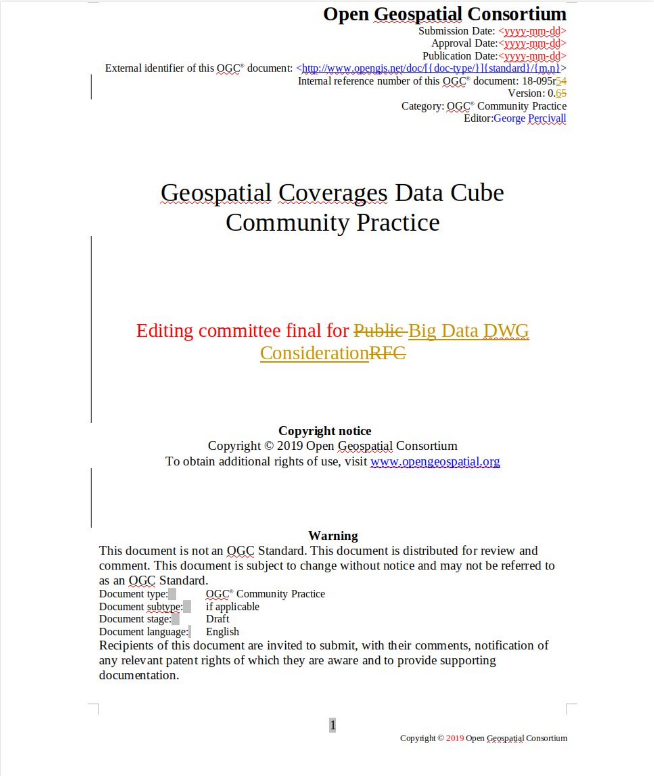
Earth Observation Data Cubes

- No common definition, mostly a multidimensional array with spatial, temporal, and/or spectral / variable dimensions
- Example: Four dimensional regular raster data cube (*referred to as* ***EODC***):
  - 4d array (x, y, time, bands)
  - Spatial axes aligned with SRS axes
  - b × t × y × x →real number
  - Single spatial reference system (SRS)
  - Cells have constant temporal duration, and spatial size



Standards?

- OGC Coverages: More general, include data cubes, point clouds, general meshes
- WCS: Accessing coverages over web services
- WCPS: Processing coverages
- WPS: generic web-based Geoprocessing
- do not solve:
    - How can I run custom Python / R / Julia scripts on the data
    - How to build data cubes from irregular satellite image collections



Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>
Approval Date:<yyyy-mm-dd>
Publication Date:<yyyy-mm-dd>
External identifier of this OGC® document: <http://www.opengis.net/doc/{doc-type/}{standard}/{m.n}>
Internal reference number of this OGC® document: 18-095r5~~4~~
Version: 0.6~~5~~
Category: OGC® Community Practice
Editor:George Percivall

Geospatial Coverages Data Cube Community Practice

Editing committee final for ~~Public~~ Big Data DWG Consideration~~RFC~~

Copyright notice
Copyright © 2019 Open Geospatial Consortium
To obtain additional rights of use, visit www.opengeospatial.org

Warning
This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Document type:	OGC® Community Practice
Document subtype:	if applicable
Document stage:	Draft
Document language:	English

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

1

Copyright © 2019 Open Geospatial Consortium

How to Create Data Cubes?

1 Understanding data products

- File organization
- Metadata Extraction

2 Define Data Cube Parameters

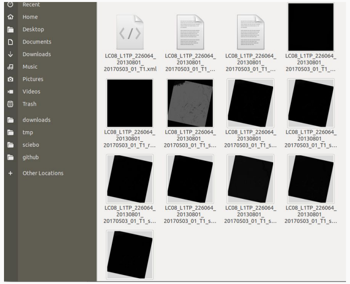
- Extent
- Cell size
- SRS
- Bands

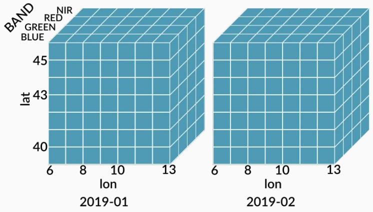
3 Image Warping

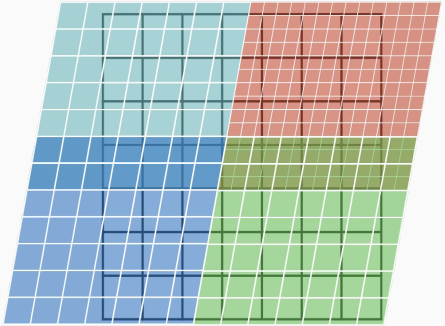
- Reprojection
- Rescaling
- Cropping
- Resampling

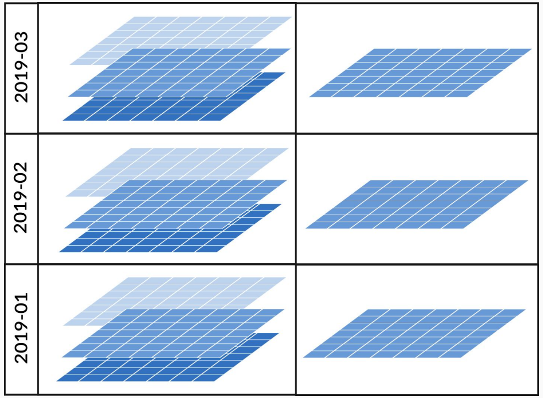
4 Temporal Aggregation

- Combine values at the same data cube cell from multiple images









- Available tools: GDAL + R / Python / ...
- Data cube creation is needed, but not for free: computations, loss of information
- There is no *correct* data cube

Data Cube Implementations

- Earth System Data Cube (ESDC) [1]
- Open Data Cube [2]
- gdalcubes [3]
- EuroDataCube [4]
- (Google Earth Engine [5])
- ...

OPEN DATA CUBE

EURO DATA CUBE

GDAL CUBES

[1] Mahecha, M. D., Gans, F., Brandt, G., Christiansen, R., Cornell, S. E., Fomferra, N., ... & Donges, J. F. (2020). Earth system data cubes unravel global multivariate dynamics. *Earth System Dynamics*, 201-234.
[2] Lewis, A., Oliver, S., Lymburner, L., Evans, B., Wyborn, L., Mueller, N., ... & Wu, W. (2017). The Australian geoscience data cube—foundations and lessons learned. *Remote Sensing of Environment*, *202*, 276-292.
[3] Appel, M., & Pebesma, E. (2019). On-demand processing of data cubes from satellite image collections with the gdalcubes library. *Data*, *4*(3), 92.
[4] https://eurodatacube.com/
[5] Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. Remote sensing of Environment, 202, 18-27.

Differences of Data Cube Implementations

- Pre-grid vs. on-the fly creation
- dimensionality (n-dimensional vs. fixed dimensions)
- availability of data cube operations and processing algorithms
- availability of programming language interfaces
- support for irregular, labeled dimensions
- flexibility to let users define target cube parameters
- support to run user-defined functions over subsets / dimensions of data cubes
- vector vs. raster data cubes

gdalcubes

A C++ library and R package for on-the-fly creation and processing of Earth observation data cubes

gdalcubes

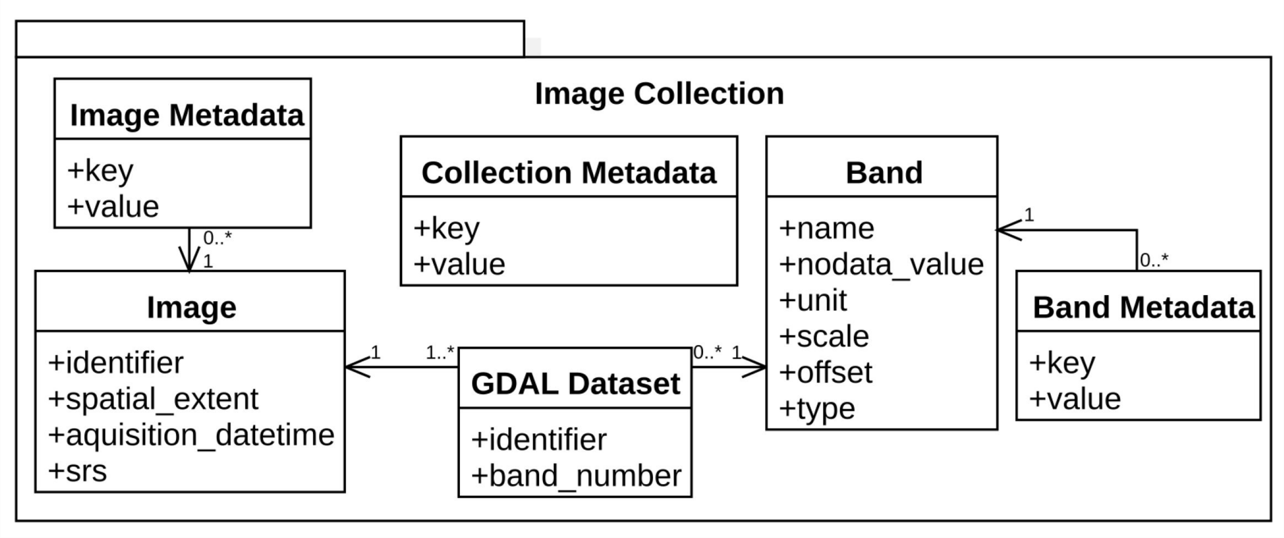
- Open source C++ library (interfacing GDAL, netCDF, SQLite, and a few other libraries)
- R package (available on CRAN) → more accessible to data scientists / statisticians
- Objective: Make complex analysis of satellite image collections easier, more interactive, and faster
- Knows how to read collections from Sentinel-2, Landsat, MODIS, PlanetScope and more
- Four-dimensional raster data cube model (space, time, band / variable)
- On-the-fly creation and processing of data cubes

gdalcubes

- Original imagery is *indexed* only → no 2nd copy of the data
- Flexibility to apply the same analysis on different spatiotemporal resolution, larger areas, …
- Data cubes are lazily evaluated (when users call plot / save results)
- Processing works chunk-wise in memory
- Data cube creation may include the application of masks (clouds, QA, …)

Basic Concepts: Image Collection

An image collection is a set of n images, where images contain m variables or spectral bands. Band data from one image share a common spatial footprint, acquisition date/time, and spatial reference system but may have different pixel sizes. Technically, the data of bands may come from one or more files, depending on the organization of a particular data product. [1]



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

Basic Concepts: Collection Formats

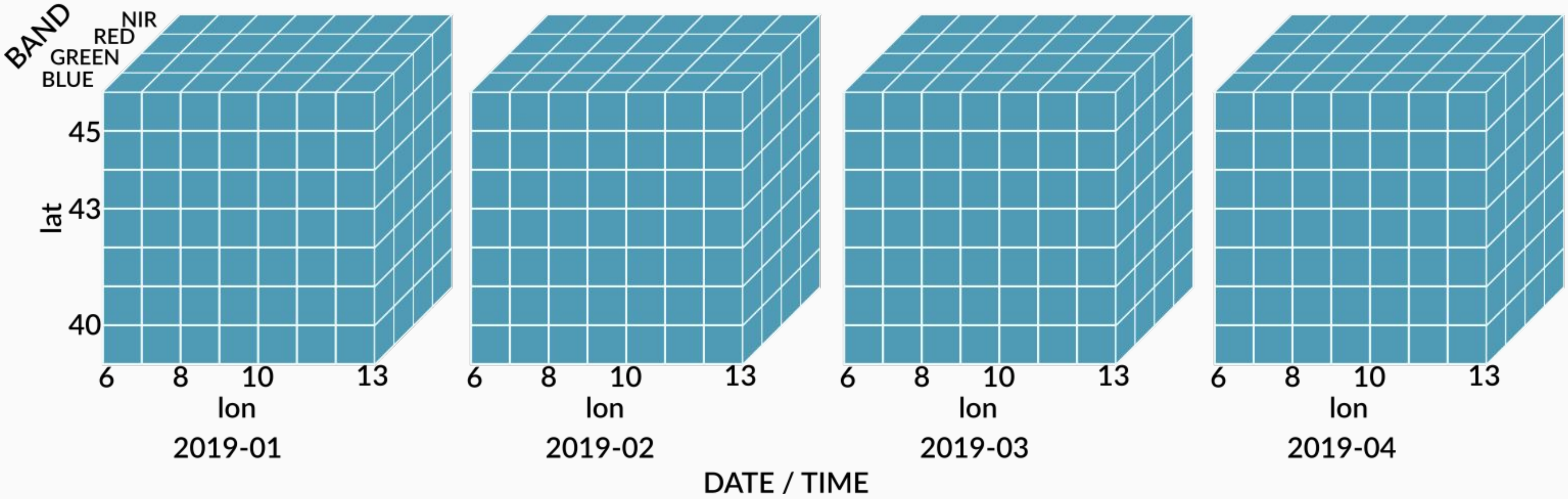
- EO data products are distributed in very different formats (see e.g. Sentinel-2 vs. Landsat 8)
- Collection format = set of rules how relevant metadata can be extracted from products
- Includes definition of bands
- Internally, relatively simple JSON format, repository available at https://gdalcubes.github.io/docs/collection_formats.html
- Extensible, documented at https://github.com/gdalcubes/collection_formats

Basic Concepts: Data Cube View

- Data cube view = Cube geometry + creation options
  - spatiotemporal extent,
  - spatiotemporal pixel sizes / counts
  - Coordinate reference system
  - spatial resampling, temporal aggregation
  -
- Does not include bands / variables, independent from particular data products
- Spatial resampling methods: (see gdalwarp) near, bilinear, cubic, cubicspline, lanczos, average, mode, max, min, med, q1, q3, sum
- Temporal aggregation methods: min, mean, median, max, sum

Basic Concepts: Regular raster data cube

- Regular raster data cube = image collection + data cube view
- or result of further data cube processing



Example

Minimal R example:

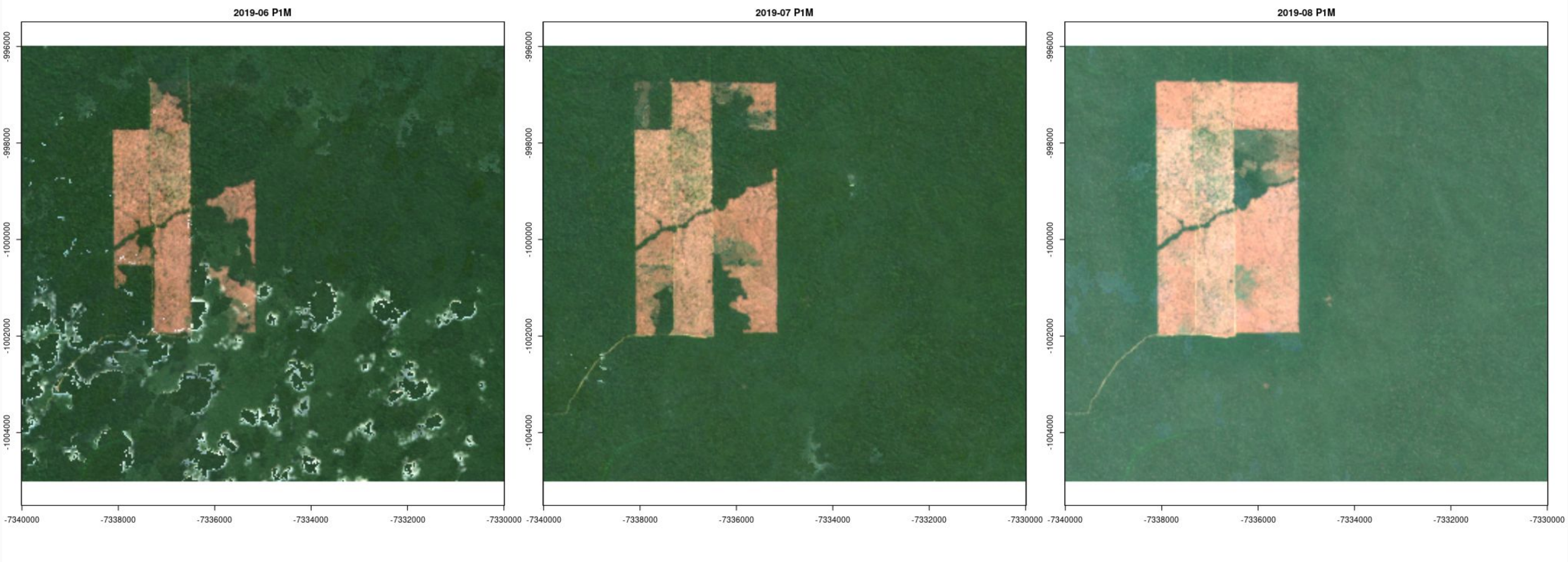
```r
library(gdalcubes)

# 1. Build image collection
files = list.files("~/Sentinel2_data", pattern = ".zip", full.names = TRUE)
S2.col = create_image_collection(files, "Sentinel2_L2A")

# 2. Define target data cube geometry
v = cube_view(srs="EPSG:3857", extent = list(left=-7340000, right=-7330000,
              bottom=-1005000, top=-995999, t0="2019-06", t1="2019-08"),
              dx = 30, dy = 30, dt = "P1M", resampling = "average",
              aggregation="median")

# 3. Build and plot spacetime RGB data cube
raster_cube(S2.col, v) %>%
  select_bands(c("B02","B03","B04")) %>%
  plot(rgb=3:1, zlim=c(0,1200))
```

Example



2019-06 P1M
2019-07 P1M
2019-08 P1M
-996000
-998000
-1000000
-1002000
-1004000
-7340000
-7338000
-7336000
-7334000
-7332000
-7330000

Operations on Data Cubes

`apply_pixel`	Apply arithmetic expressions on band values per pixel
`fill_time`	Fill missing values by simple time-series interpolation
`filter_pixel`	Filter pixels based on logical expressions
`filter_geom`	Filter pixels that do not intersect with a given input geometry
`join_bands`	Combine bands of two or more identically shaped input data cubes
`reduce_space`	Apply a reducer function over time slices of a data cube
`reduce_time`	Apply a reducer function over individual pixel time series
`select_bands`	Select a subset of a data cube’s bands
`window_time`	Apply a moving window reducer of kernel over individual pixel time series

Extraction from data cubes

`query_timeseries`	Extract one or more pixel time series
`query_points`	Export data cube values at irregular spacetime points
`zonal_statistics`	Compute summary statistics over polygons

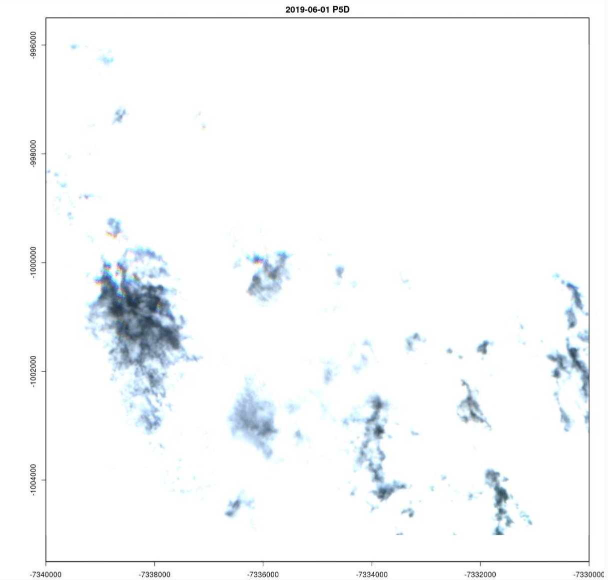
Toy example:

- zonal statistics (min, mean, median, max NDVI) over ~50k cadastral districts ("Fluren”)
- One year of Sentinel-2 L2A data (> 500 GB)
- Aggregated monthly, 10m spatial resolution

Data Cube Export / Visualization

`write_ncdf`	Export a data cube as a single netCDF file
`write_tif`	Export a data cube as a collection of GeoTIFF / COG files
`plot`	Plot data cubes
`animate`	Create animations from data cube time slices

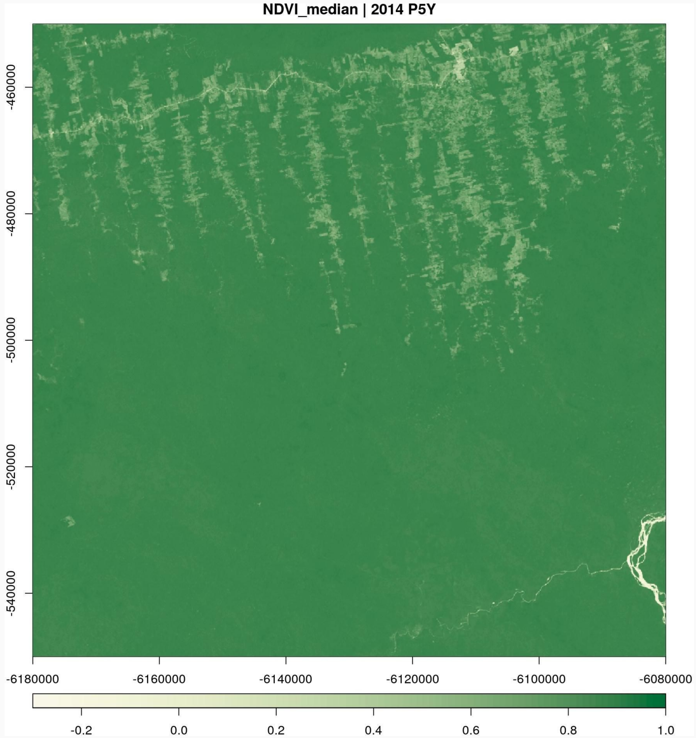
- Export supports compression and packing (using smaller integer types instead of double precision floating point)
- Ready to be processed in external software / other R packages



Chaining Data Cube Operations

Operations can be chained, creating a *processing graph*:

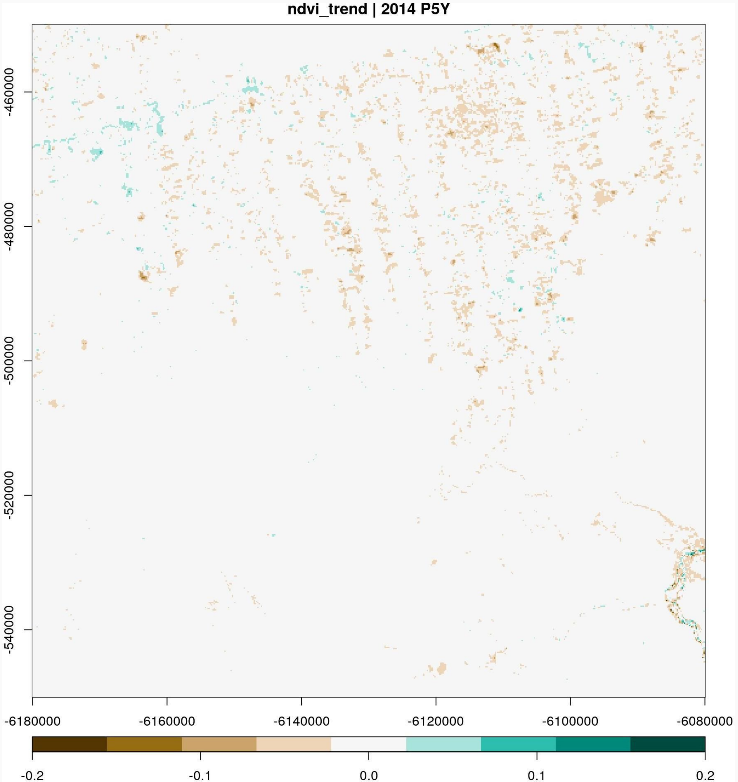
```
raster_cube(L8.col, v.subarea.100m) %>%
  select_bands(c("B04","B05")) %>%
  apply_pixel("(B05-B04)/(B05+B04)", names="NDVI") %>%
  reduce_time("median(NDVI)") %>%
  plot(col=ndvi.col, nbreaks=100, zlim=c(-0.3,1),
       key.pos = 1)
```



User-defined Functions

User-defined functions can be streamed to time series / spatial slices of a data cube:

```
raster_cube(L8.col, v) %>%
  select_bands(c("B04","B05")) %>%
  apply_pixel("(B05-B04)/(B05+B04)","NDVI") %>%
  reduce_time(names=c("ndvi_trend"),
    FUN=function(x) {
      z = data.frame(t=1:ncol(x),
                     ndvi=x["NDVI",])
      coef(lm(ndvi ~ t, z))[2]
  }) %>%
  plot(key.pos=1, col=col.trend,
       nbreaks=10, zlim=c(-0.2,0.2))
```



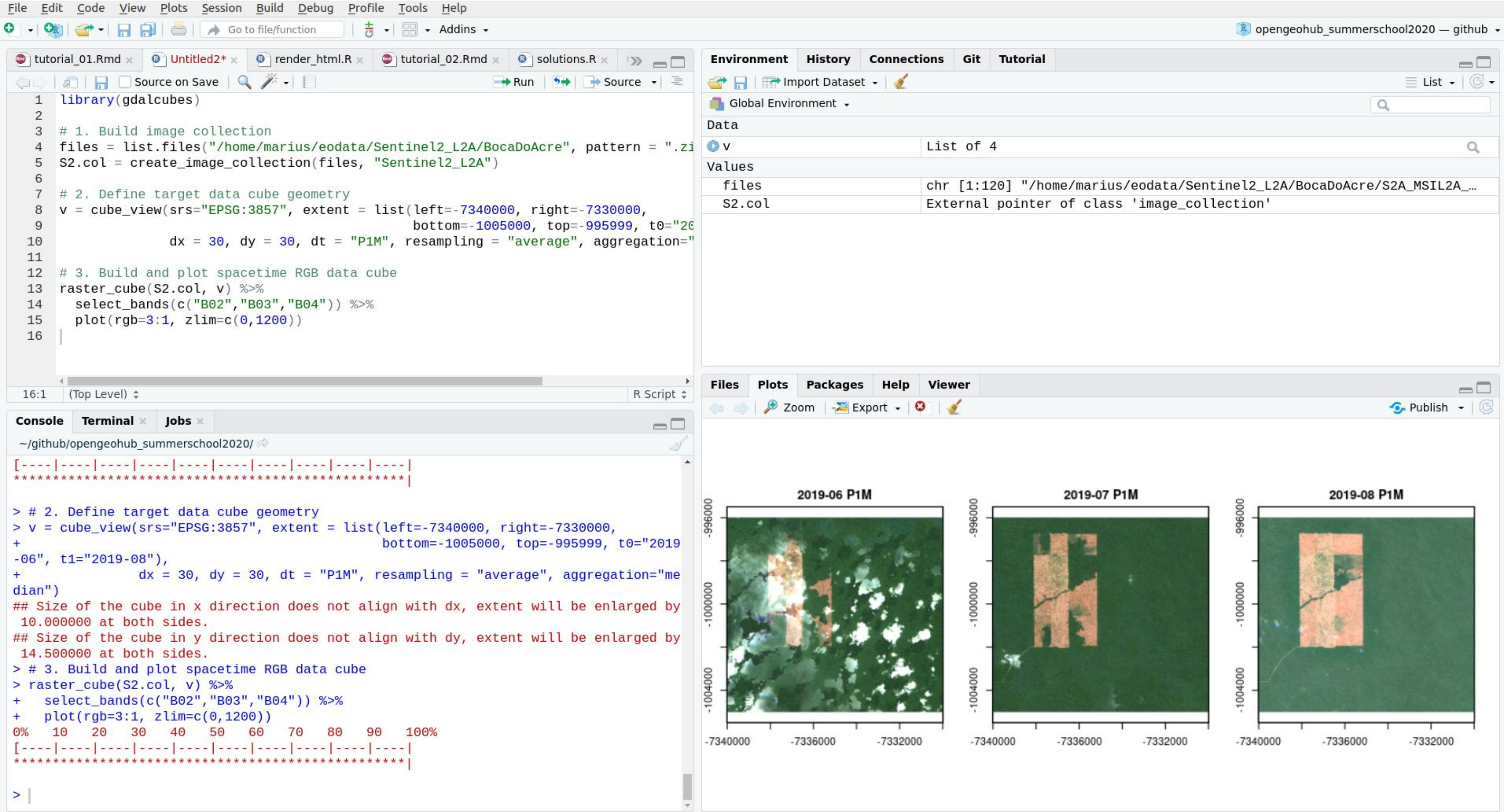
Incremental Method Development

Data cubes are created on-the-fly; it is straightforward to go from low resolution experiments to applying algorithms on high resolution.

Example: *Derive median RGB values over all pixel time series at different spatial resolution of a collection with approx. 90 GB compressed Sentinel 2 L2A images:*

Pixel size	**Computation time**
300m x 300m	40 seconds
50m x 50m	26 minutes
10m x 10m	2 hours

Hands on with gdalcubes



Discussion and Outlook

- Complex analysis becomes possible with user-defined functions (using functions from any available R package)
- Performance of data cube creation depends on data formats, depending on the method, this may or may not be relevant
- Data model is limited to 4 dimensions (x, y, t, band / variable) and requires orthorectified images. Data such as Sentinel-1 or Sentinel-5P need to be preprocessed before they can be used with gdalcubes. The stars package is much more flexible in these cases.
- Can be used in cloud computing environments, but some work is needed (see discussion later)

Discussion and Outlook

- Lots of ideas for the future work, including
  - openEO backend implementation
  - Data cube export / processing as services (WCS, WMTS, WPS)
  - Easier cloud deployment
  - Python interface?
  - …

- Get in touch if you have further ideas, questions, or want to contribute in any other way…

Applications

Examples how EO data cubes can be used for time series processing and for combining data from different EO missions.

(1) Combined NDVI data cube from Sentinel-2, Landsat-8, and MODIS

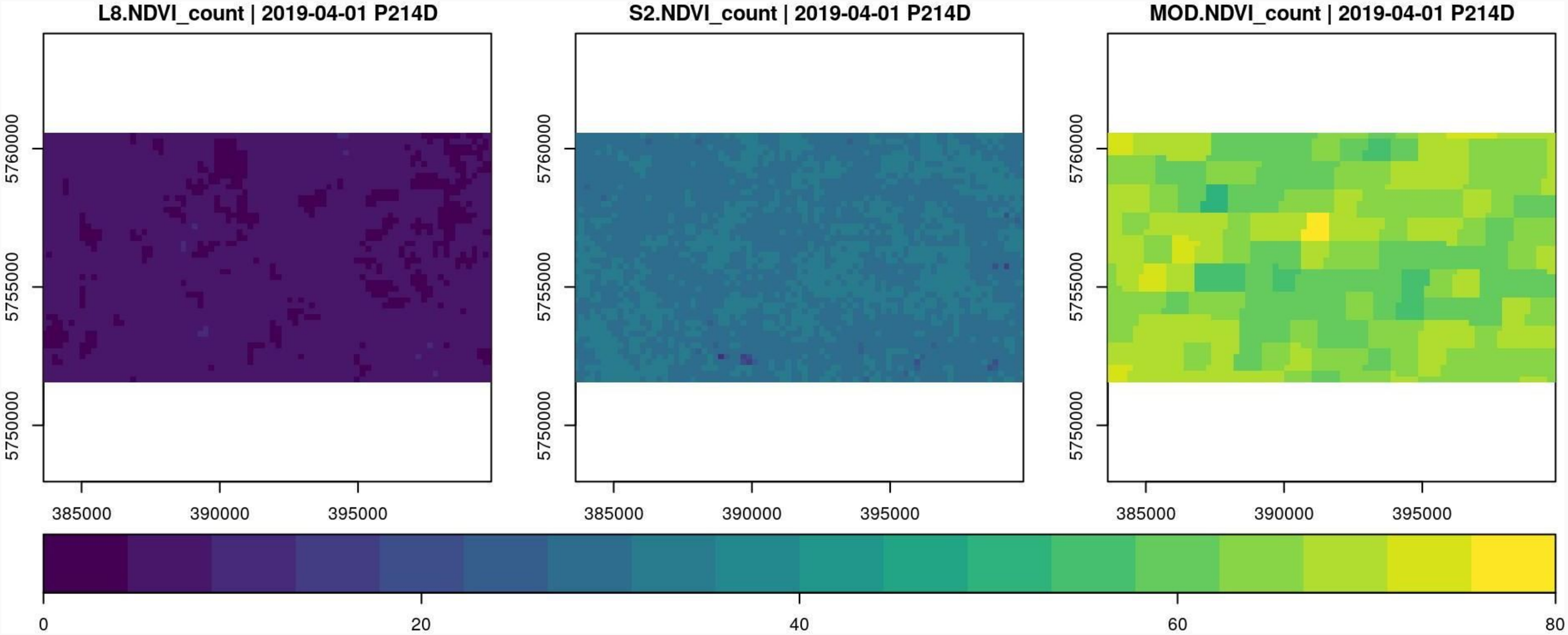
Idea: Monitoring the vegetation with NDVI as target variable, using as much information as possible

Dataset	Spatial Resolution (NDVI bands)	Temporal Resolution	File Format
Sentinel-2 Level 2A	10m	5 days	.jp2
Landsat 8 surface reflectance	30m	16 days	GeoTIFF
MODIS MOD09GA	500m	daily	HDF4

(Optimistic) Approach:

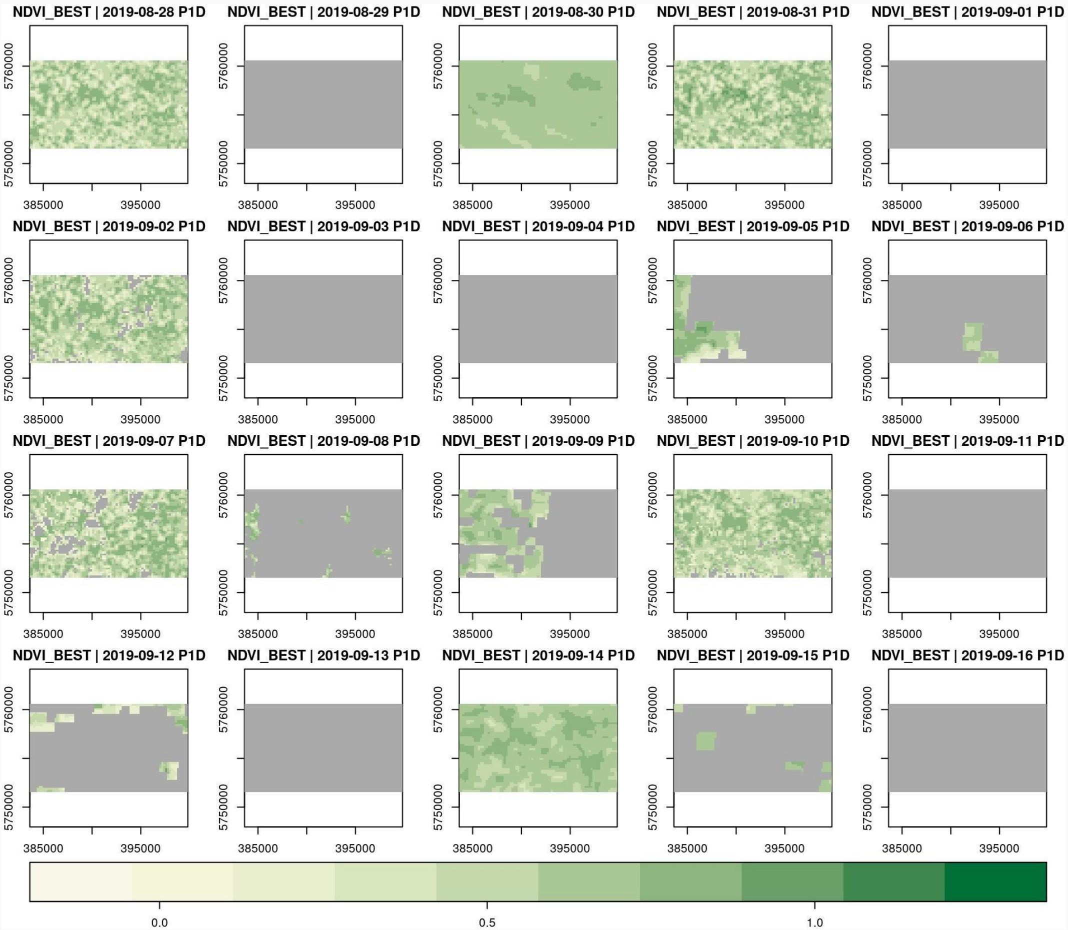
1. Build three separate NDVI data cubes with the same *geometry*
2. Stack data cubes
3. Iterate over individual pixel time series and select best available measurement for each t

(1) Combined NDVI data cube from Sentinel-2, Landsat-8, and MODIS



(1) Combined NDVI data cube from Sentinel-2, Landsat-8, and MODIS

Result (subset, low resolution):



(2) Combined analysis of vegetation, precipitation, and soil moisture

Idea: Analyze correlations between different variables

Dataset	Spatial Resolution	Temporal Resolution	File Format
MODIS 13A2 [1]	1km	16 days	HDF4
GPM_3IMERGDF [2]	0.1°	daily	GeoTIFF
ESA CCI soil moisture [3,4,5]	0.25°	daily	netCDF

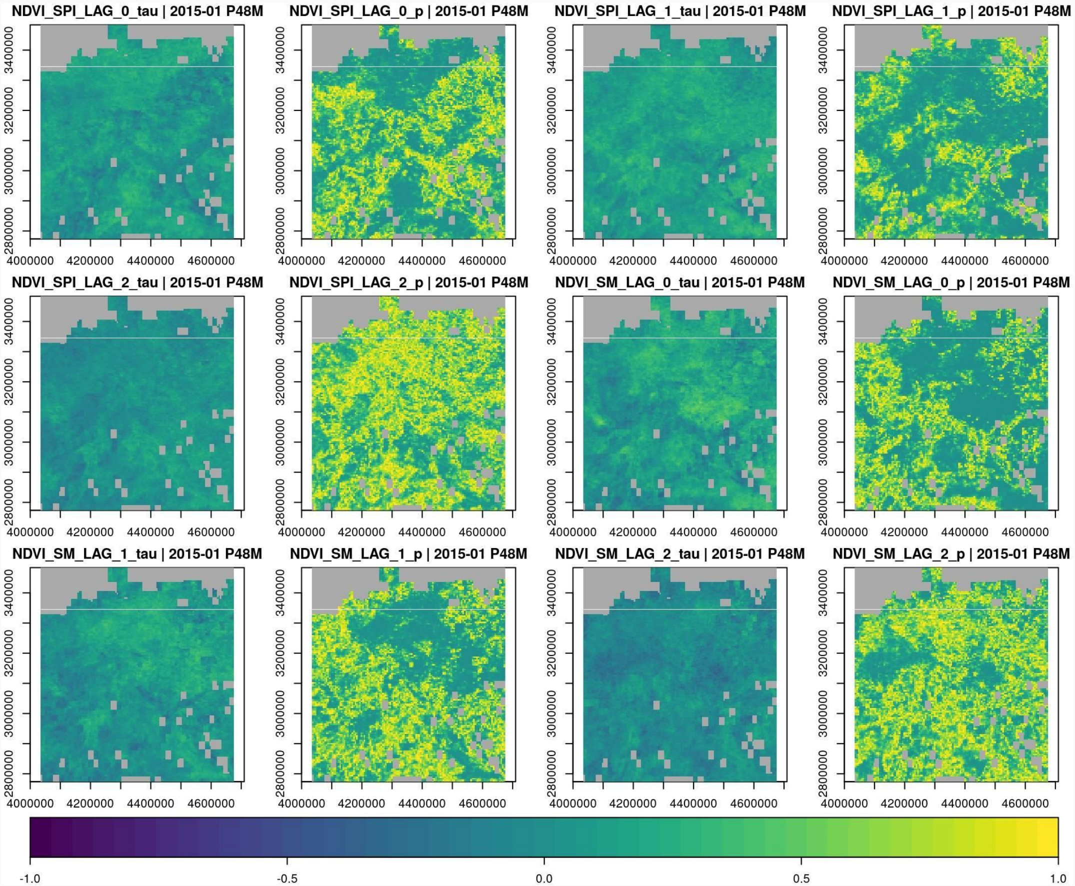
(Optimistic) Example:

1. Build three separate monthly data cubes with the same *geometry*
2. Preprocess individual data cubes:
   a. Calculate NDVI + SM anomalies by time series decomposition
   b. Calculate standardized precipitation index
3. Apply tests for temporally lagged correlation

[1] https://lpdaac.usgs.gov/products/mod13a2v006
[2] https://disc.gsfc.nasa.gov/datasets/GPM_3IMERGDF_06/summary
[3] https://doi.org/10.5194/essd-11-717-2019
[4] https://doi.org/10.1016/j.rse.2017.07.001
[5] https://doi.org/10.1109/TGRS.2017.2734070

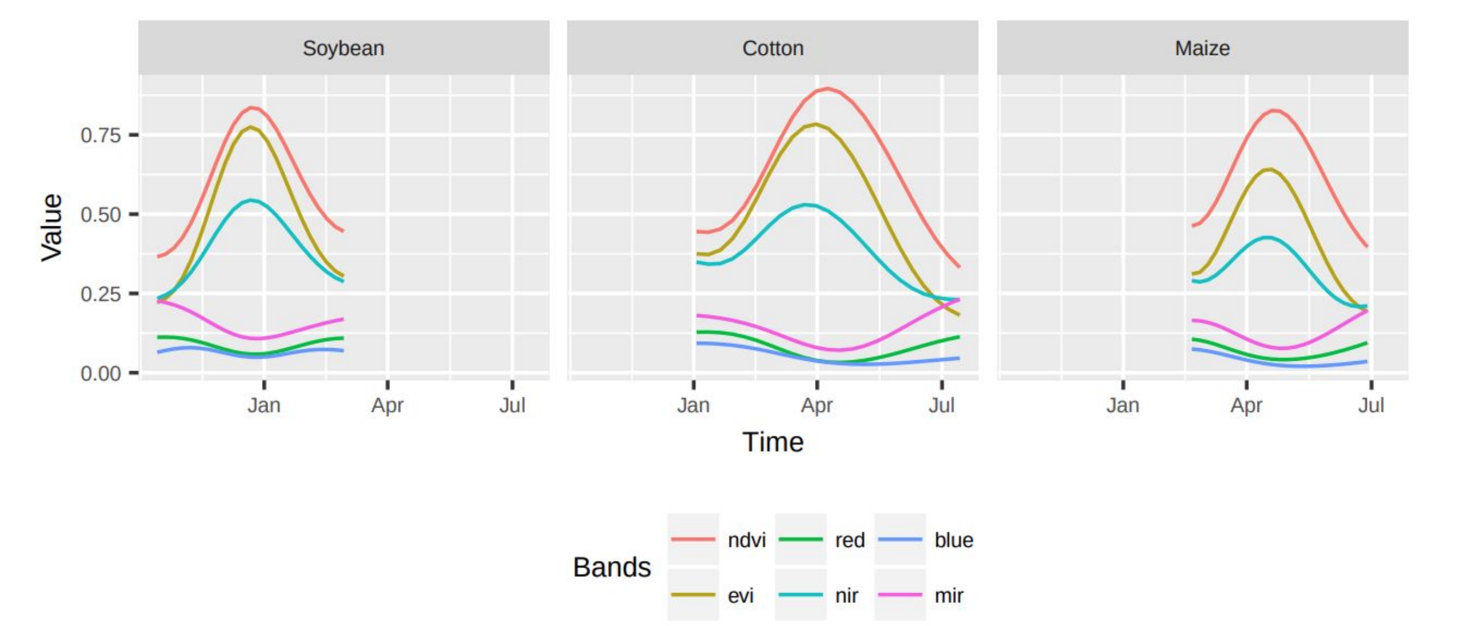
(2) Combined analysis of vegetation, precipitation, and soil moisture

Result (subset, low resolution):



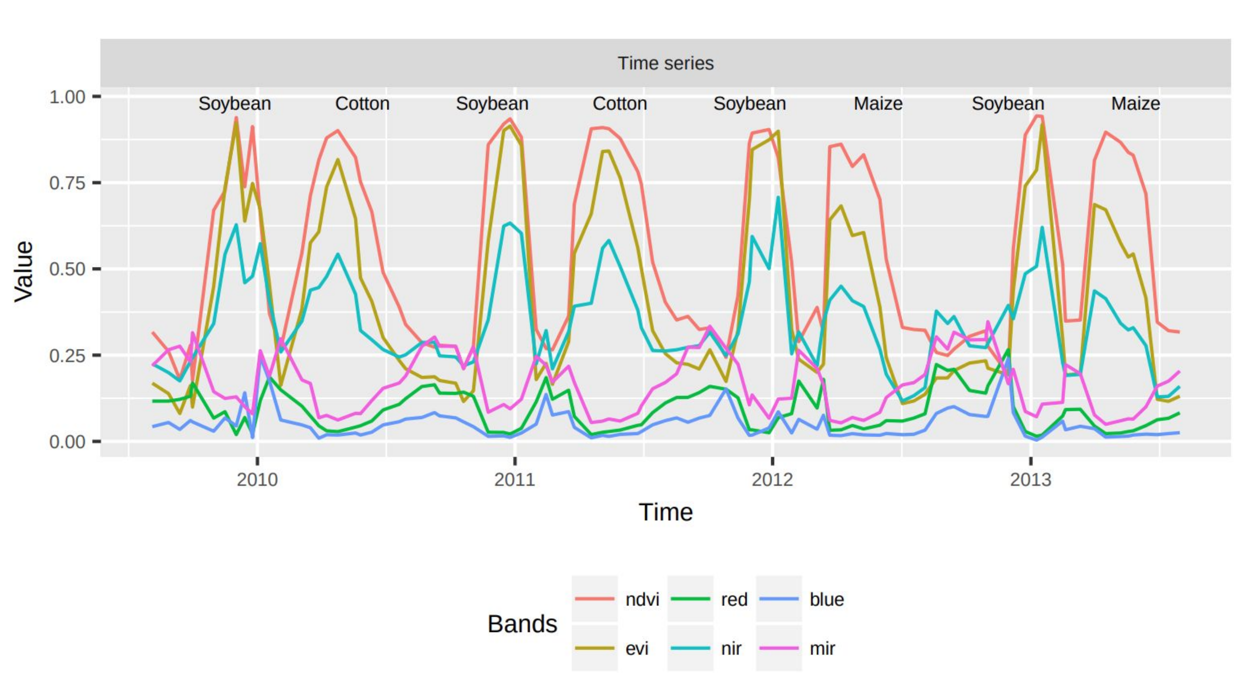
(3) Crop classification with dynamic time warping

- Idea: match multivariate time series to provided patterns
- Method is available as an R package [1]



[1] Maus, V., Câmara, G., Appel, M., & Pebesma, E. (2019). dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R. *Journal of Statistical Software, 88*(5), 1 - 31. doi:http://dx.doi.org/10.18637/jss.v088.i05

(3) Crop classification with dynamic time warping



In progress: Application on Landsat data from 1997 on national scale

Maus, V., Câmara, G., Appel, M., & Pebesma, E. (2019). dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R. *Journal of Statistical Software, 88*(5), 1 - 31. doi:http://dx.doi.org/10.18637/jss.v088.i05

Discussion and Outlook

More Applications of Data Cubes

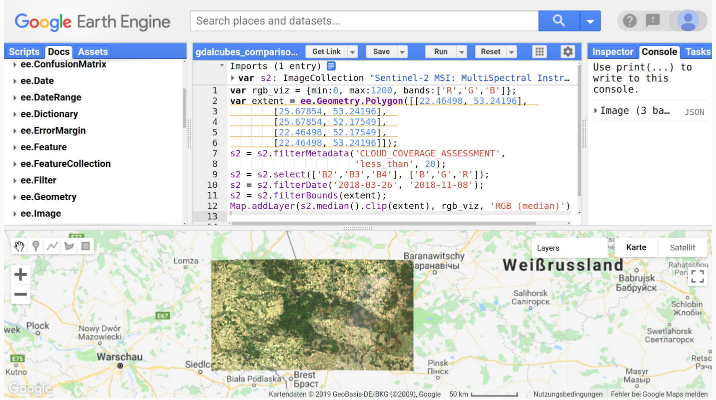
- Complex machine learning methods, e.g. finding causal effects in multivariate EO time series (see e.g. [1])
- Spatiotemporal Statistics: Dealing with spatiotemporal autocorrelations in the data (e.g. [2])
- In general: time series analysis, combining datasets with different properties., processing large areas
- Does *analysis ready* include the content of pixels?
- Methods need to consider nonoptimal preprocessing (atmospheric correction, cloud masks, …)

[1] Runge, J., Bathiany, S., Bollt, E. *et al.* Inferring causation from time series in Earth system sciences. *Nat Commun* 10, 2553 (2019).

[2] Appel, M., & Pebesma, E. (2020). Spatiotemporal multi-resolution approximations for analyzing global environmental data. Spatial Statistics, 100465.

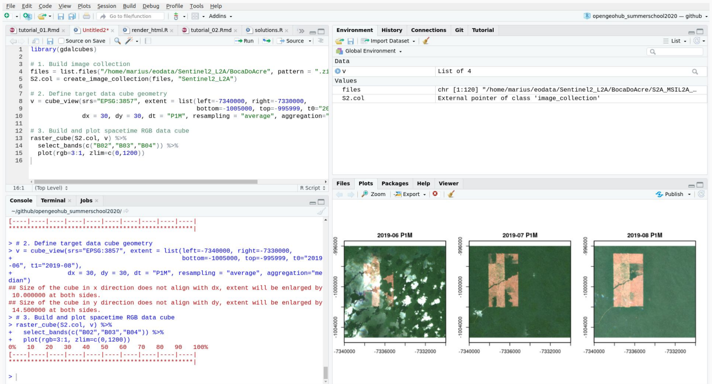
Many platforms for EO analysis = Difficult Choice

Use Google Earth Engine (GEE), other cloud services, or do the processing locally / at the university's computing center?



Criteria to decide where to run EO data analysis:

- Availability of data and algorithms
- Costs
- Programming language interfaces[1]
- Effort needed for re-implementations
- Trust in long-term existence
- Needed data volume / scalability
- Data cube vs. file interface



1 addressed in OpenEO project https://openeo.org/

STAC + COGs in the cloud

STAC
SpatioTemporal
Asset Catalog

- SpatioTemporal Asset Catalog → Easy data discovery with modern technology

- STAC API: RESTful WFS3 compatible API for searching STAC items

COG
CLOUD OPTIMIZED
GEOTIFF

- Cloud-native image format, supporting HTTP range requests → efficient extraction of subsets, and overview images

- More and more provides / platforms offer access to COGs

Summary

- Earth observation data cubes facilitate the extraction of information from large irregular satellite image collections
- Data cubes make it easy to combine data from different satellite-based EO missions
- Construction of data cubes from satellite image collections is complex and involves spatial resampling, reprojection, and temporal aggregation
- gdalcubes is an open-source library and R package to facilitate the analysis of satellite image collections

Contact: marius.appel@uni-muenster.de

Further reading:

- https://www.mdpi.com/journal/data/special_issues/EODC
- https://github.com/appelmar/opengeohub_summerschool2020