https://shorturl.at/uXNUp

Optimizing Global Time Series ML Predictions

From Modeling to Production Deployment

Leandro Parente

leandro.parente@opengeohub.org

https://opengeohub.org

+ OGH crew

Global Pasture Watch

A consortium of research partners developing global grassland and livestock monitoring data for impact

OpenGeoHUB
Connect · Create · Share · Repeat

LAPIG
Laboratório de Processamento de Imagens e Geoprocessamento

iDiv

GLAD Global Land Analysis & Discovery

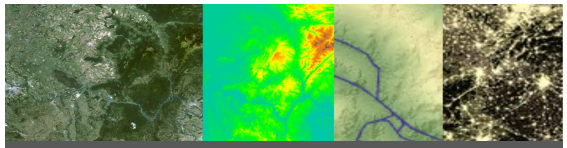
Cornell University

IIASA

WORLD RESOURCES INSTITUTE

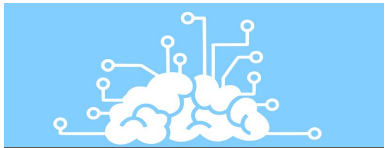
Land & Carbon Lab

\+ *User communities*



Earth Observation / Gridded data

Reference samples

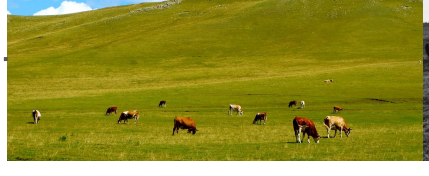


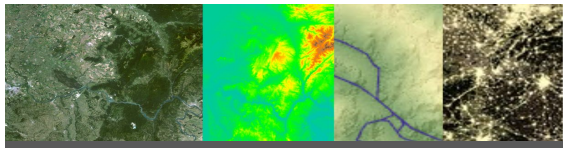
Machine learning

#	Mapping product	Temporal resolution	Spatial resolution	Pixel value
1	**Grassland dynamics maps** (2000–2022+)	Annual	30-m	Two class of grass
2	**Livestock density maps** (2000–2022+)	Annual	1-km	Number of animals per hectare
3	**Short vegetation height maps** (2000–2022+)	Annual	30-m	Canopy height in meters
4	**Gross primary productivity** maps (2000–2022+)	Bi-monthly	30-m	Kg of carbon per m2

Feedback loop

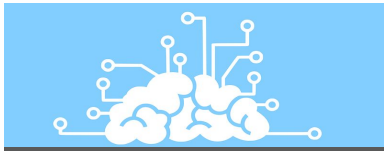
Use Cases





Earth Observation / Gridded data

Reference samples

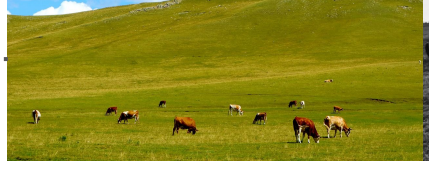


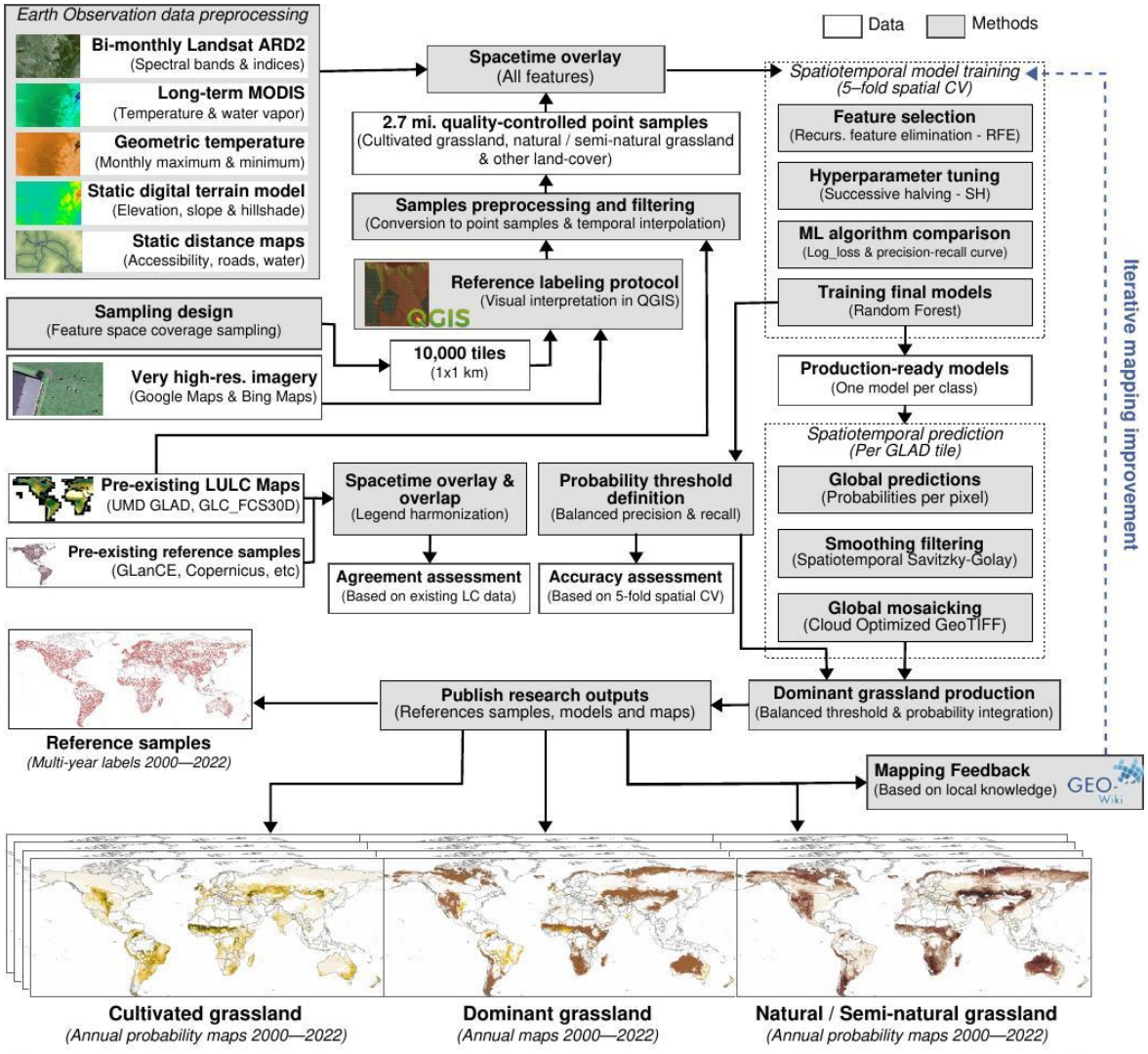
Machine learning

#	Mapping product	Temporal resolution	Spatial resolution	Pixel value
1	**Grassland dynamics maps** (2000–2022+)	Annual	30-m	Two class of grass
2	**Livestock density maps** (2000–2022+)	Annual	1-km	Number of animals per hectare
3	**Short vegetation height maps** (2000–2022+)	Annual	30-m	Canopy height in meters
4	**Gross primary productivity** maps (2000–2022+)	Bi-monthly	30-m	Kg of carbon per m2

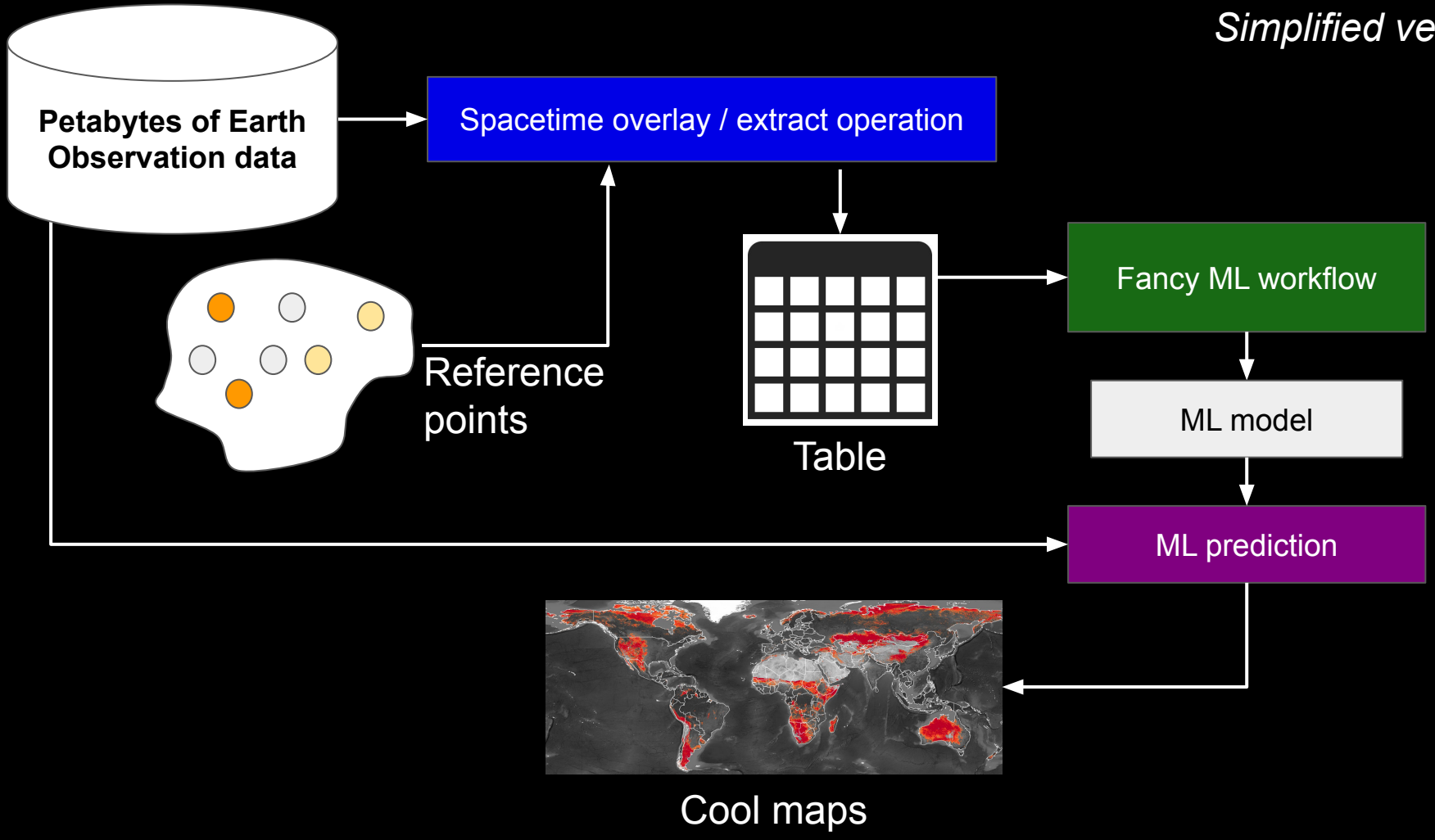
Feedback loop

Use Cases

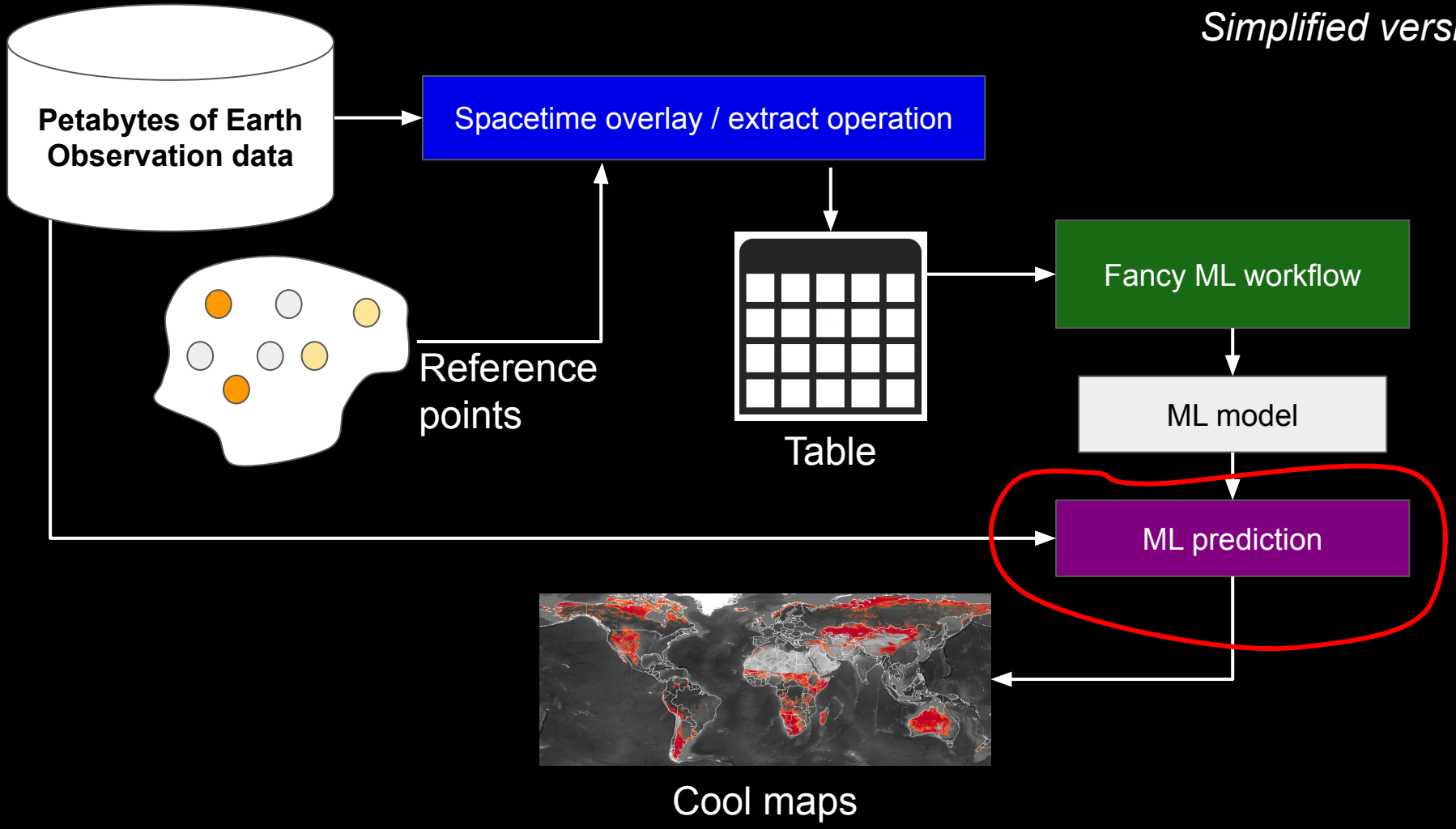



Data
Methods
Earth Observation data preprocessing
Bi-monthly Landsat ARD2
(Spectral bands & indices)
Long-term MODIS
(Temperature & water vapor)
Geometric temperature
(Monthly maximum & minimum)
Static digital terrain model
(Elevation, slope & hillshade)
Static distance maps
(Accessibility, roads, water)
Spacetime overlay
(All features)
2.7 mi. quality-controlled point samples
(Cultivated grassland, natural / semi-natural grassland & other land-cover)
Samples preprocessing and filtering
(Conversion to point samples & temporal interpolation)
Reference labeling protocol
(Visual interpretation in QGIS)
QGIS
Sampling design
(Feature space coverage sampling)
10,000 tiles
(1x1 km)
Very high-res. imagery
(Google Maps & Bing Maps)
Spatiotemporal model training
(5–fold spatial CV)
Feature selection
(Recurs. feature elimination - RFE)
Hyperparameter tuning
(Successive halving - SH)
ML algorithm comparison
(Log_loss & precision-recall curve)
Training final models
(Random Forest)
Production-ready models
(One model per class)
Spatiotemporal prediction
(Per GLAD tile)
Global predictions
(Probabilities per pixel)
Smoothing filtering
(Spatiotemporal Savitzky-Golay)
Global mosaicking
(Cloud Optimized GeoTIFF)
Iterative mapping improvement
Pre-existing LULC Maps
(UMD GLAD, GLC_FCS30D)
Pre-existing reference samples
(GLanCE, Copernicus, etc)
Spacetime overlay & overlap
(Legend harmonization)
Agreement assessment
(Based on existing LC data)
Probability threshold definition
(Balanced precision & recall)
Accuracy assessment
(Based on 5-fold spatial CV)
Dominant grassland production
(Balanced threshold & probability integration)
Publish research outputs
(References samples, models and maps)
Reference samples
(Multi-year labels 2000—2022)
Mapping Feedback
(Based on local knowledge)
GEO-Wiki
Cultivated grassland
(Annual probability maps 2000—2022)
Dominant grassland
(Annual maps 2000—2022)
Natural / Semi-natural grassland
(Annual probability maps 2000—2022)

Simplified version



Simplified version

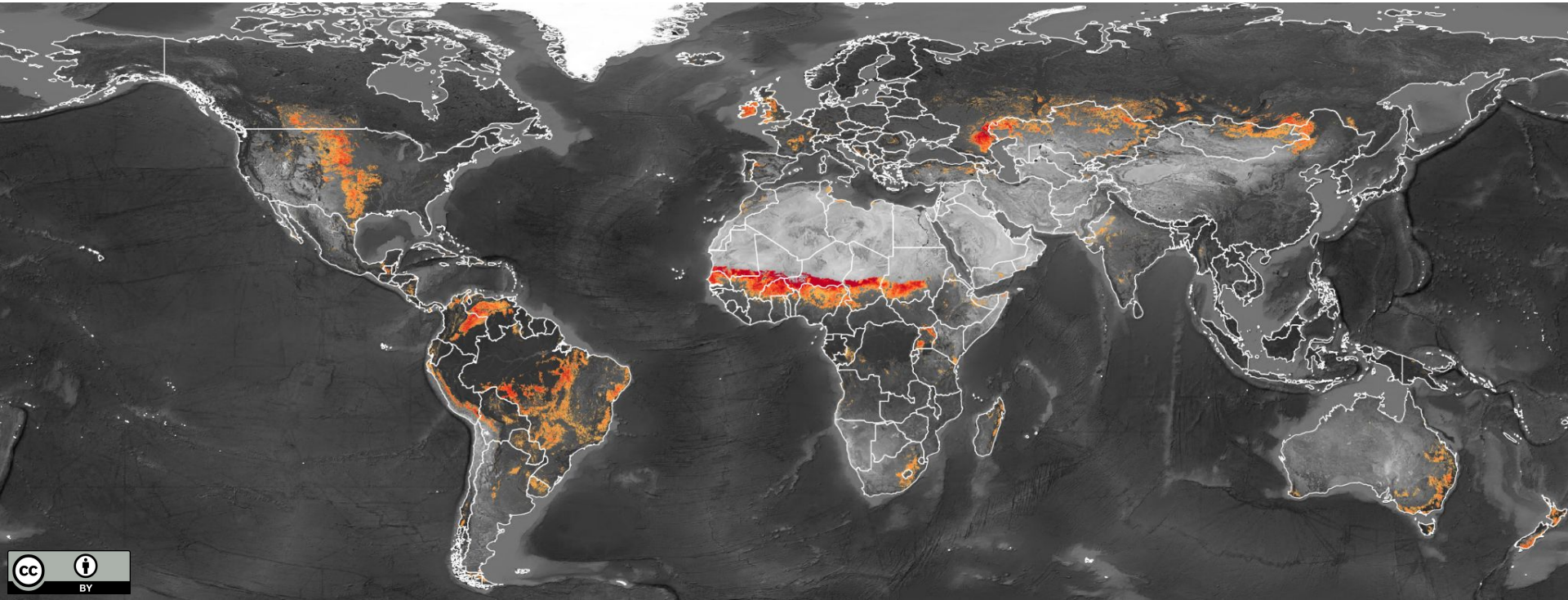


Cultivated grassland - 2022

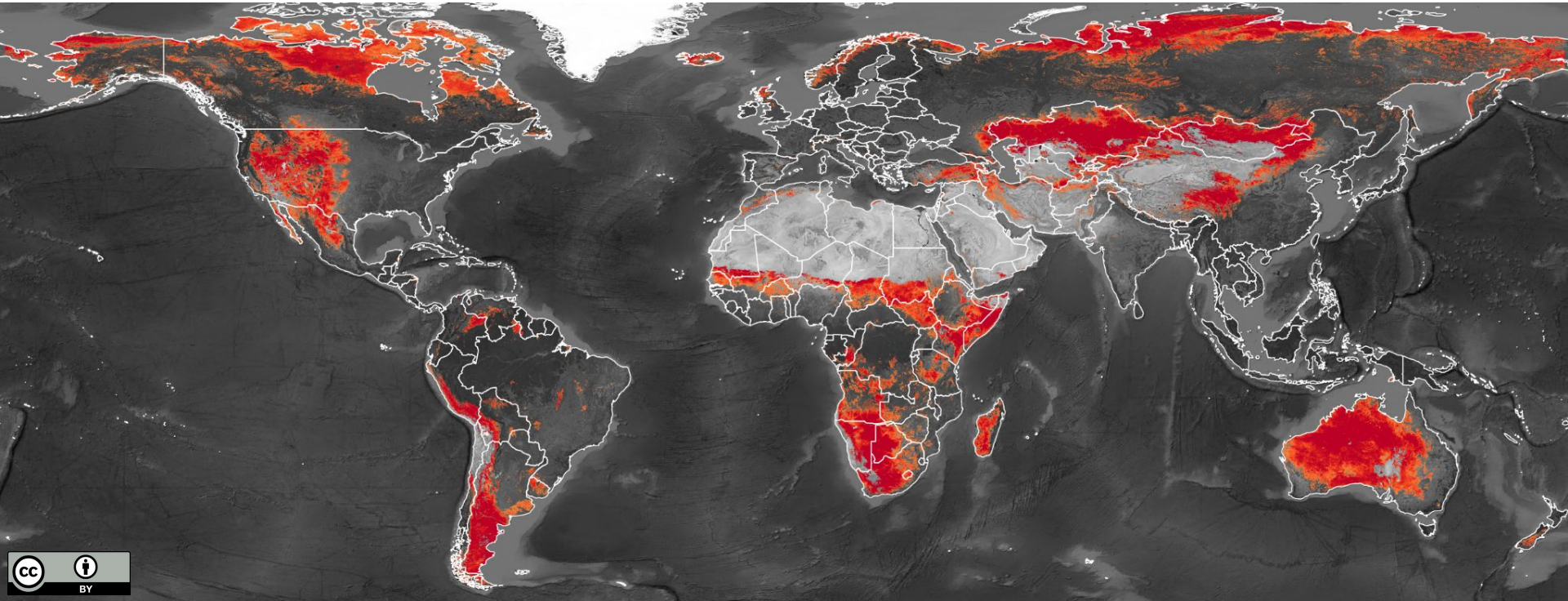
0.32

0.15 0.75

Probability threshold



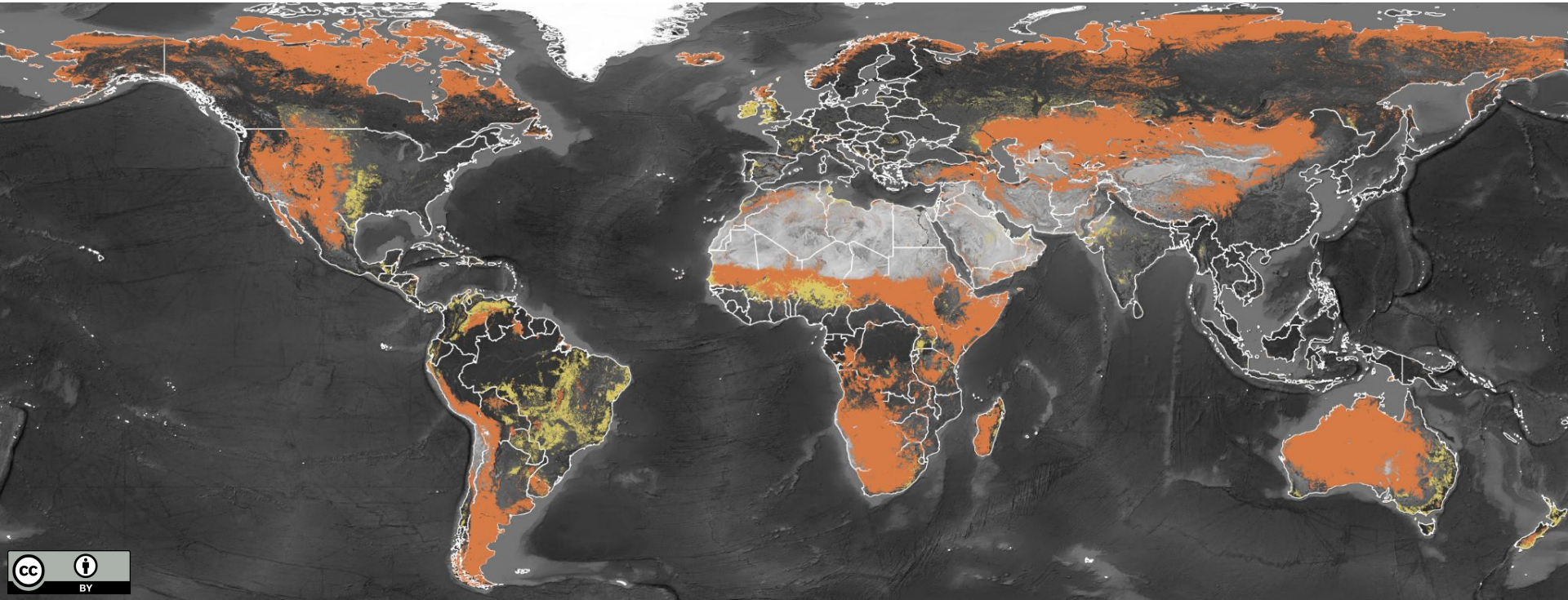
Natural / Semi-natural grassland - 2022



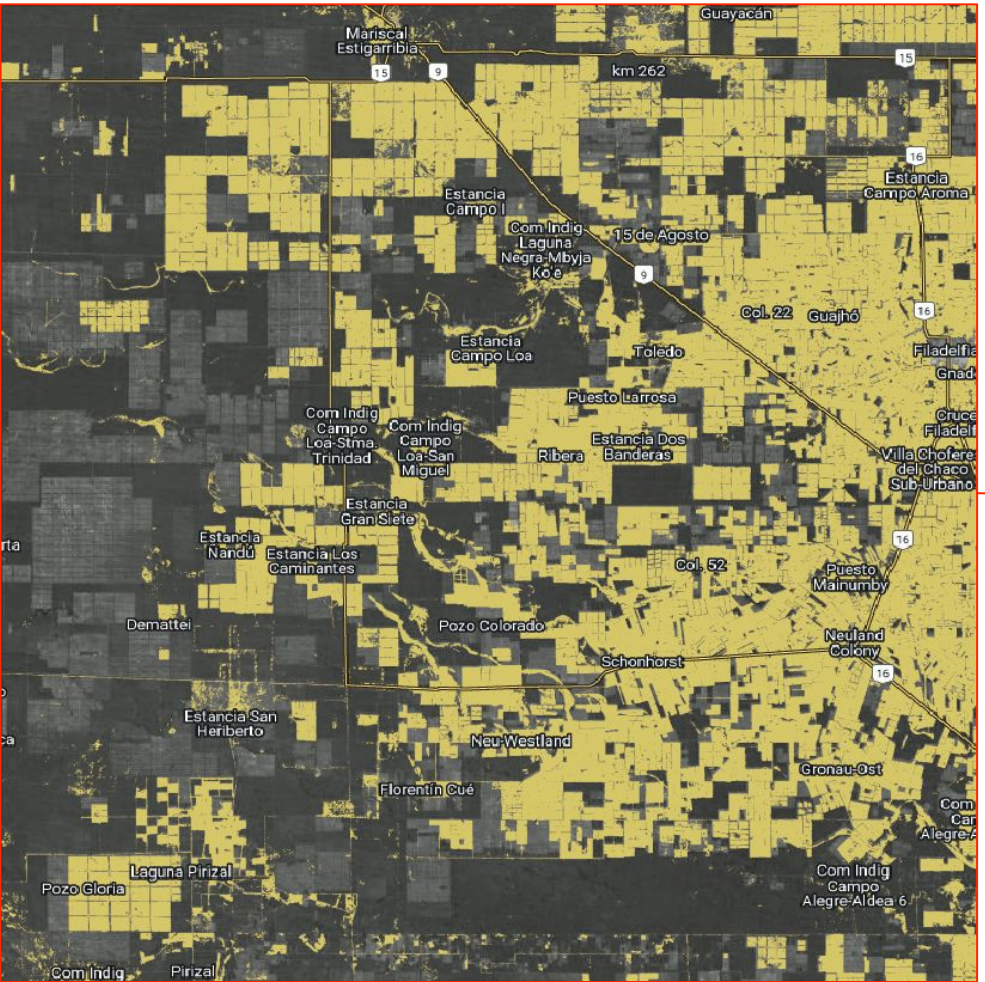
Dominant grassland - 2022

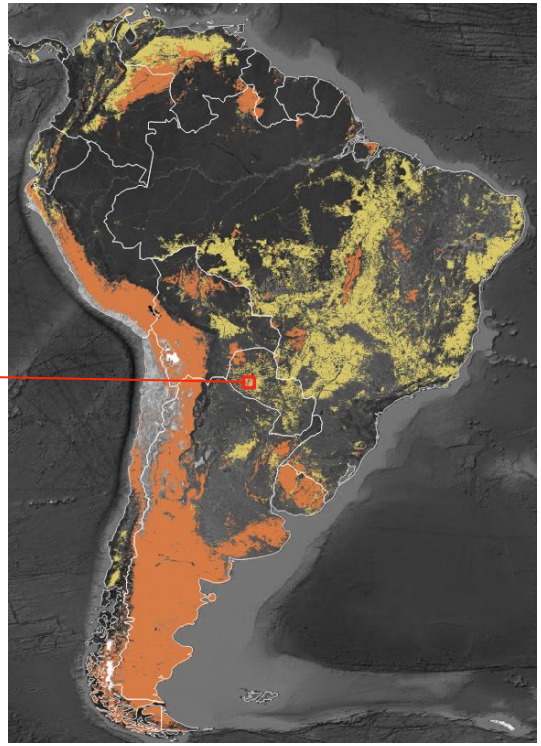
Cultivated

Natural / Semi-natural



Dominant grassland (2000–2022)





Cultivated

Natural / Semi-natural

Filadelfia, Paraguay

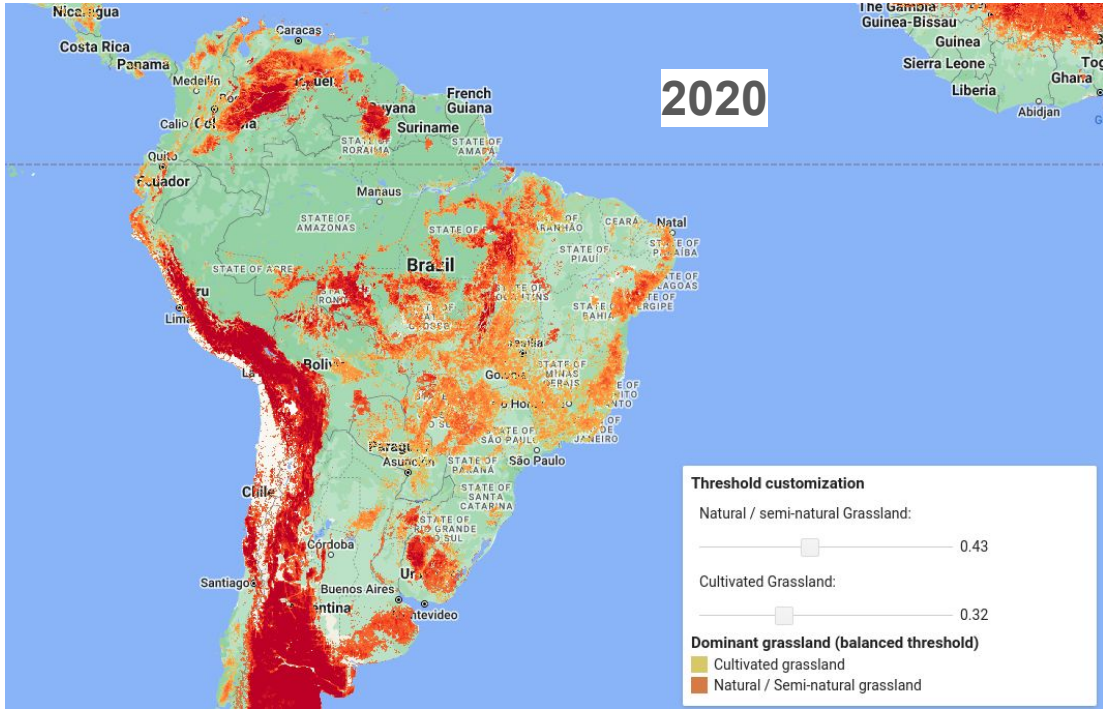
Mapping global grassland dynamics 2000—2022 at 30m spatial resolution using spatiotemporal Machine Learning

Leandro Parente[1*], Lindsey Sloat[2], Vinicius Mesquita[3], Davide Consoli[1], Radost Stanimirova[2], Tomislav Hengl[1], Carmelo Bonannella[1,4], Nathália Teles[3], Ichsani Wheeler[1], Steffen Ehrmann[5,6,8], Maria Hunter[3], Laerte Ferreira[3], Ana Paula Mattos[3], Bernard Oliveira[3], Carsten Meyer[6,7,8], Murat Şahin[1], Martijn Witjes[1,4], Steffen Fritz[5], Ziga Malek[5], and Fred Stolle[2]

[1]OpenGeoHub Foundation, Doorwerth, The Netherlands
[2]Land & Carbon Lab, World Resources Institute, Washington DC, USA
[3]Remote Sensing and GIS Laboratory (LAPIG/UFG), Goiânia, Brazil
[4]Laboratory of Geo-Information Science and Remote Sensing, Wageningen University & Research, Wageningen, The Netherlands
[5]International Institute for Applied Systems Analysis (IIASA), Laxenburg, Austria
[6]German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Leipzig, Germany
[7]Institute of Geosciences and Geography, Martin Luther University Halle-Wittenberg, Halle (Saale), Germany
[8]Institute of Biology, Leipzig University, Leipzig, Germany
[*]corresponding author(s): Leandro Parente (leandro.parente@opengeohub.org)

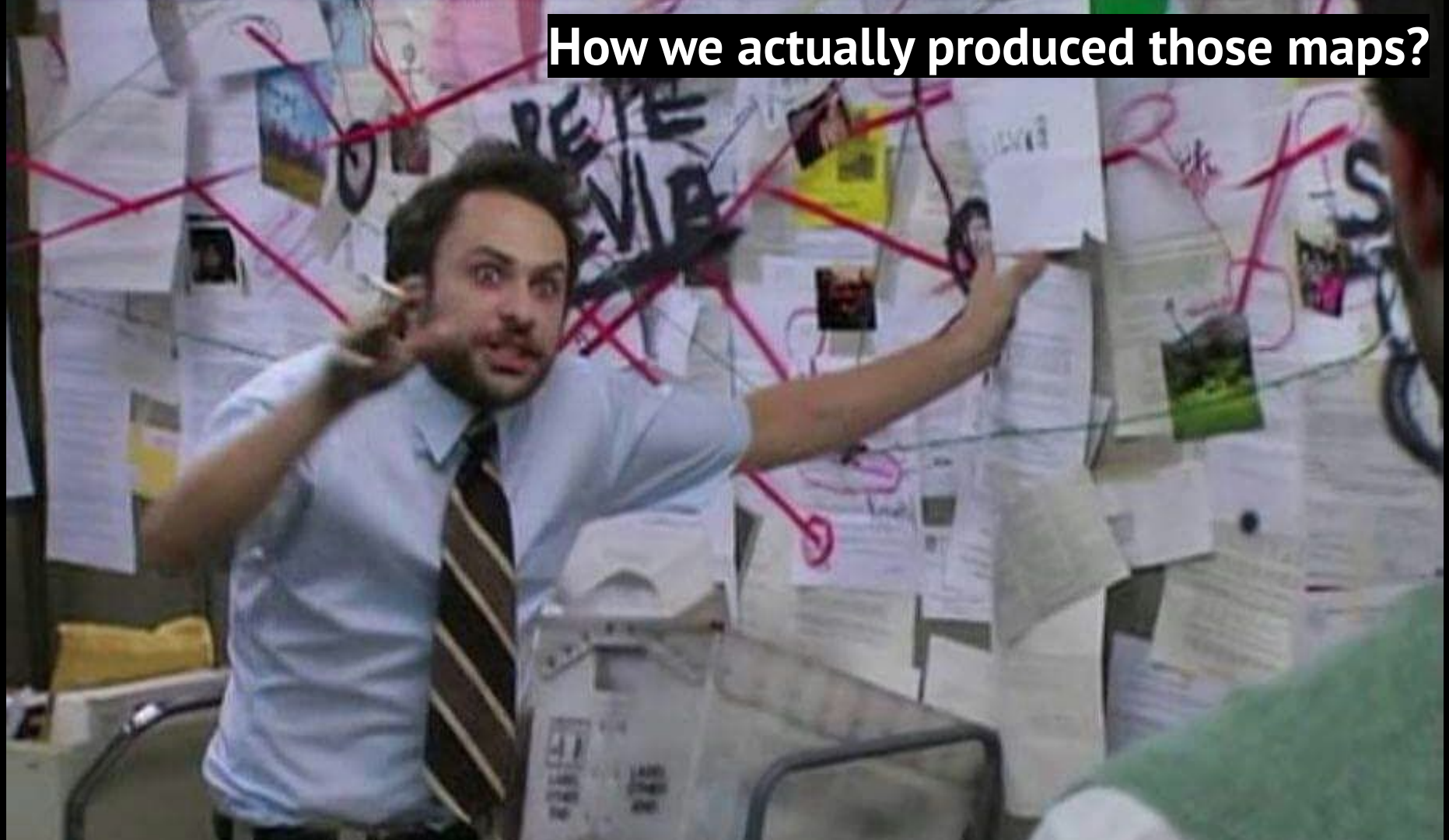
https://doi.org/10.21203/rs.3.rs-4514820/v1

https://leandro_opengeohub.users.earthengine.app/view/gpw---global-grassland-2020



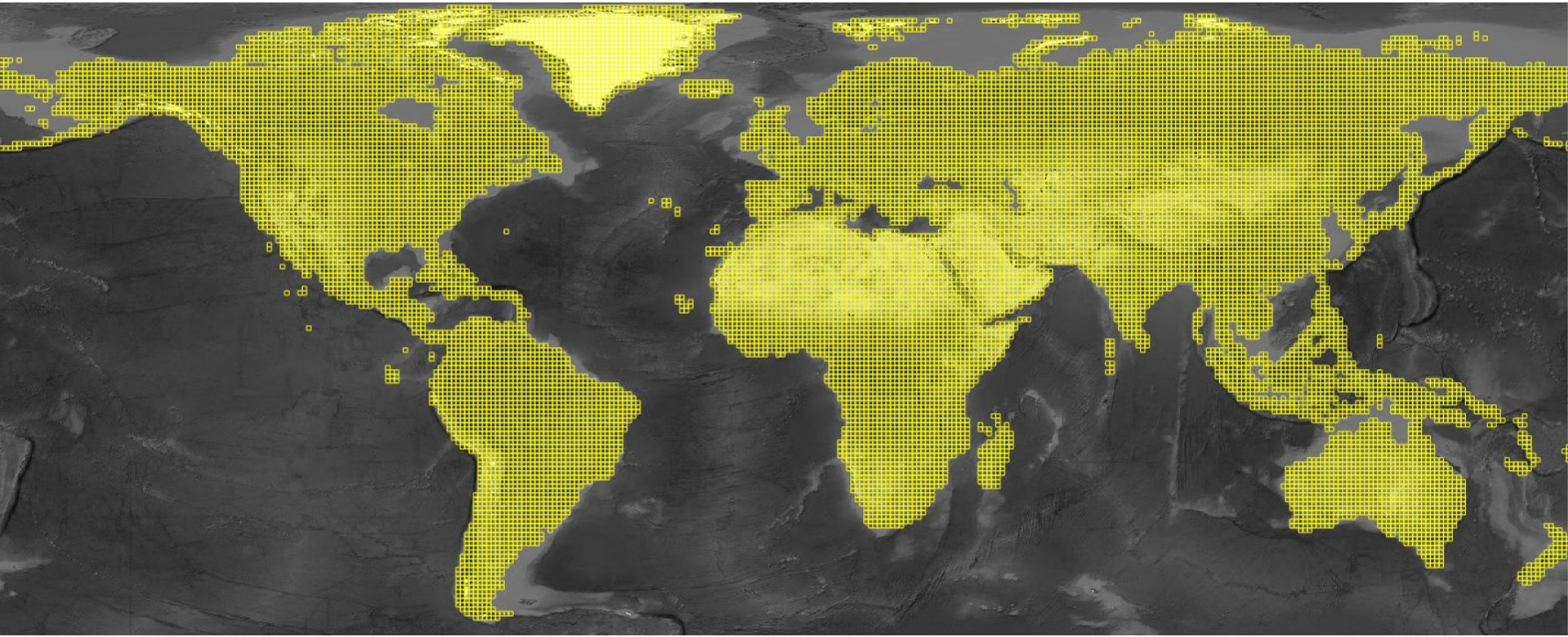
Global maps will be available though:

- Google Earth Engine
- Cloud-optimized GeoTIFF (COG)
- SpatioTemporal Asset Catalog (STAC)


How we actually produced those maps?

First, let's split the world in multiple tiles

OpenGeoHUB
Connect • Create • Share • Repeat

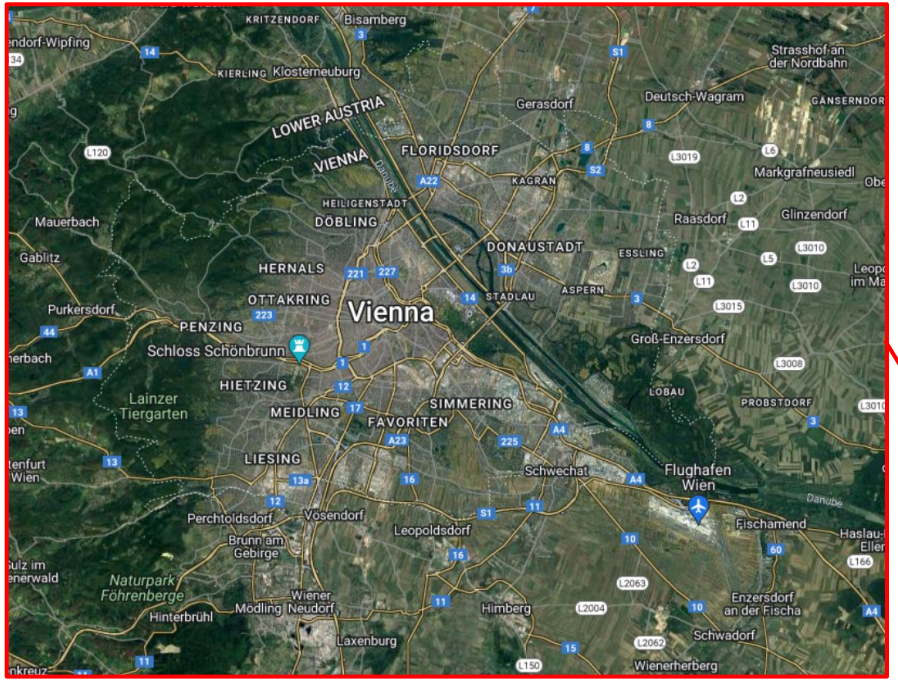


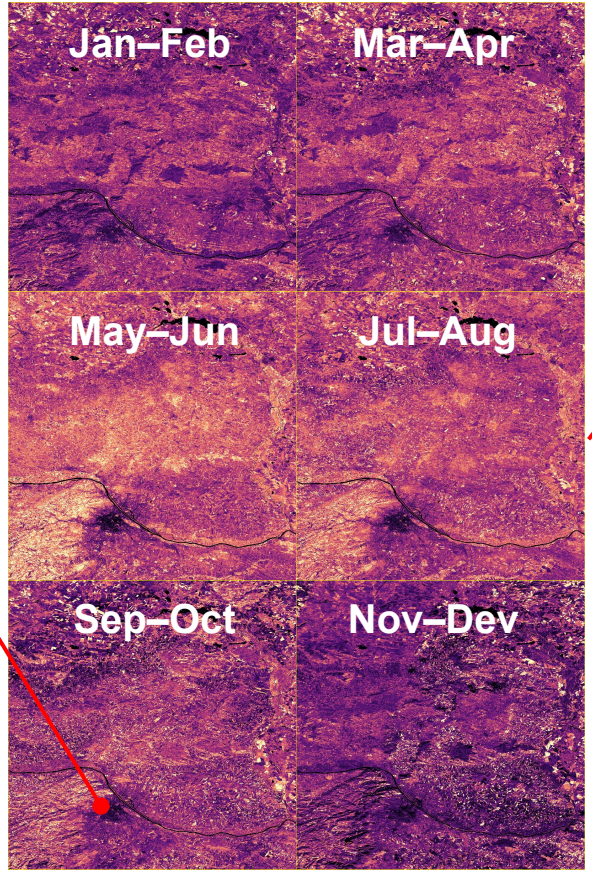
18,667 Tiles (4004 x 4004 pixels)

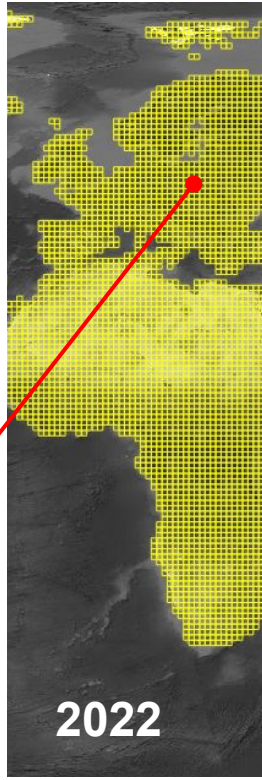
Example tile covering Viena (016E_48N)

6 dates

Near-infrared (NIR)

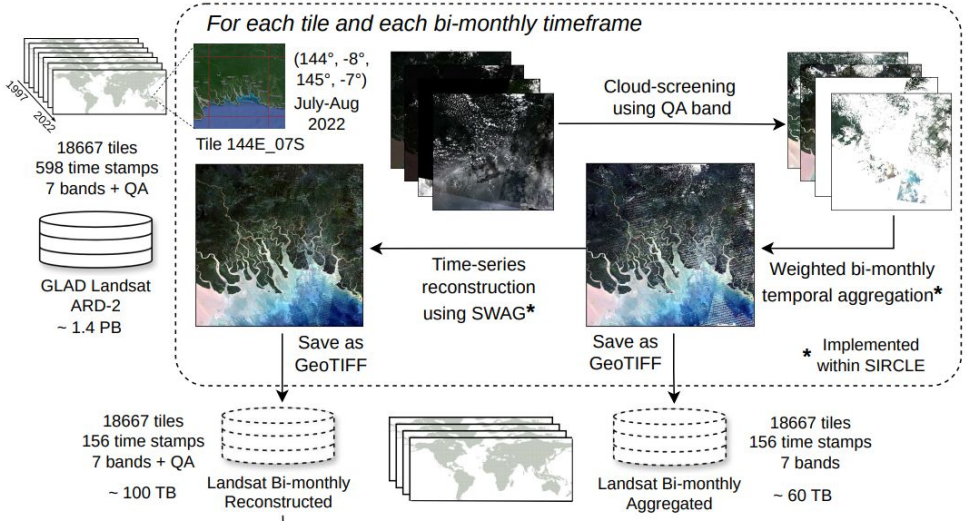






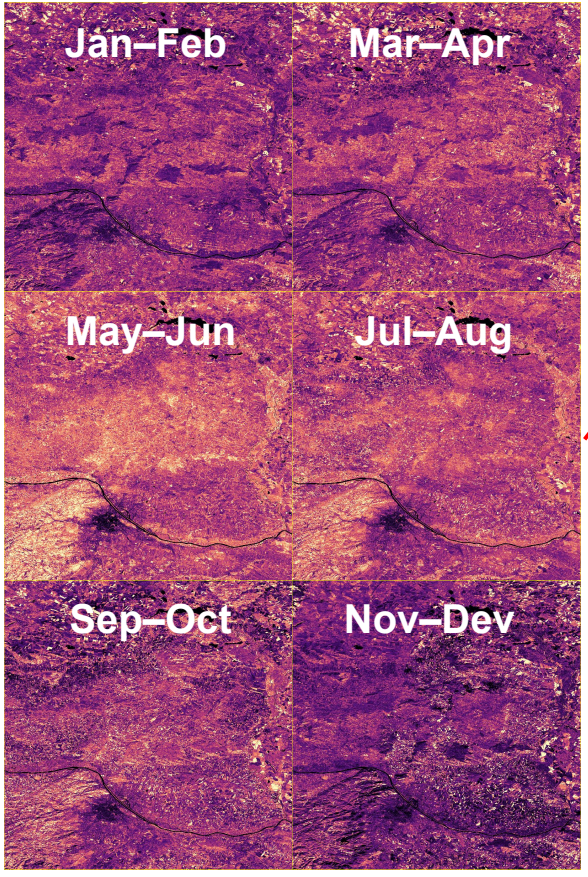
4004 x 4004 pixels

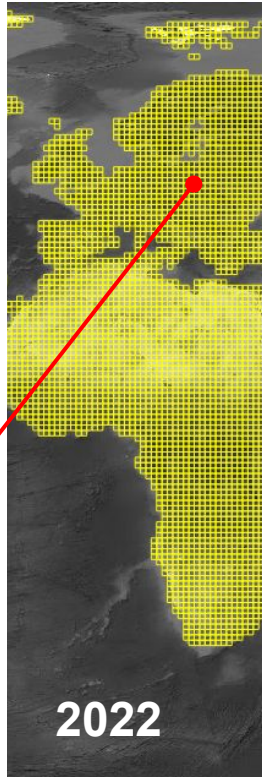
Example tile covering Viena (016E_48N)



https://doi.org/10.21203/rs.3.rs-4465582/v1

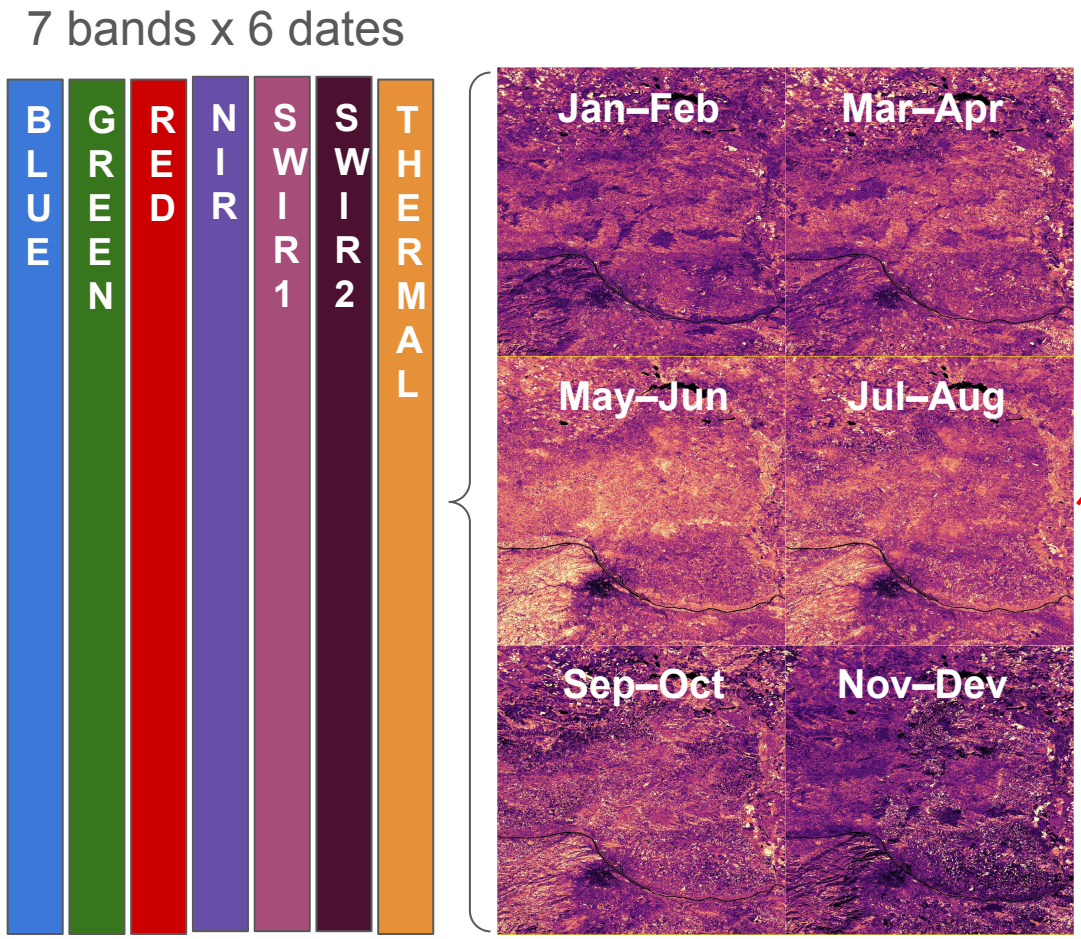
6 dates *Near-infrared (NIR)*





4004 x 4004
pixels

Temporal layers - Landsat ARD



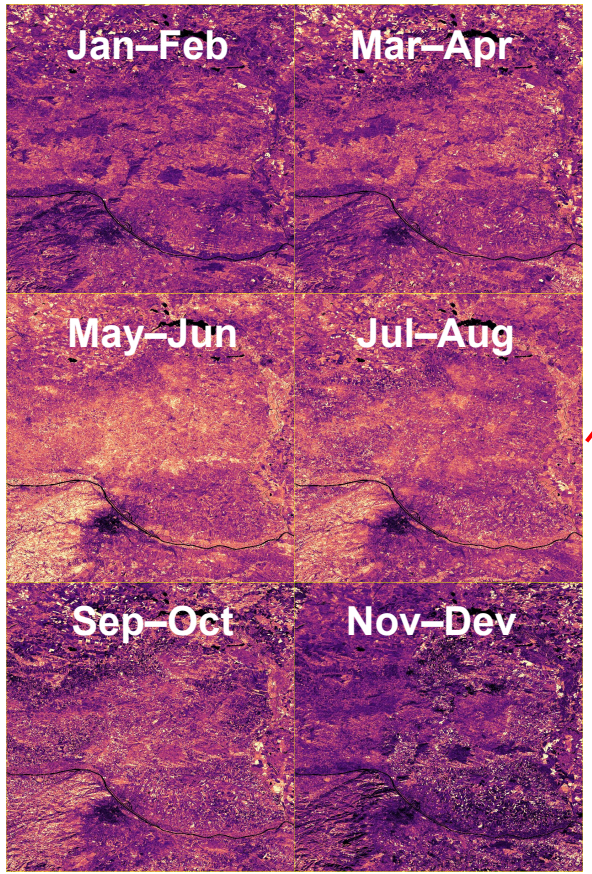
Temporal layers - Landsat ARD

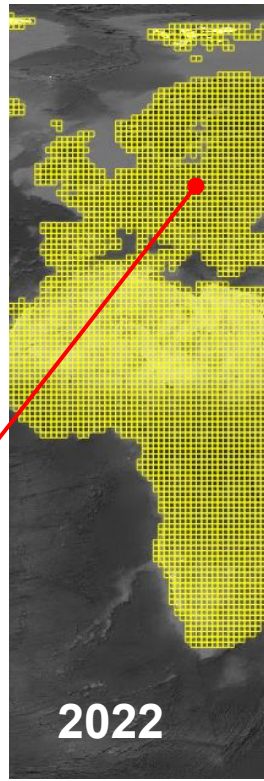
(7 indices + 7 bands) x 6 dates

NDVI
EVI
NDWI
NIRv
FAPAR
NDTI
BSI

+

BLUE
GREEN
RED
NIR
SWIR1
SWIR2
THERMAL





4004 x 4004 pixels

Temporal layers - Landsat ARD

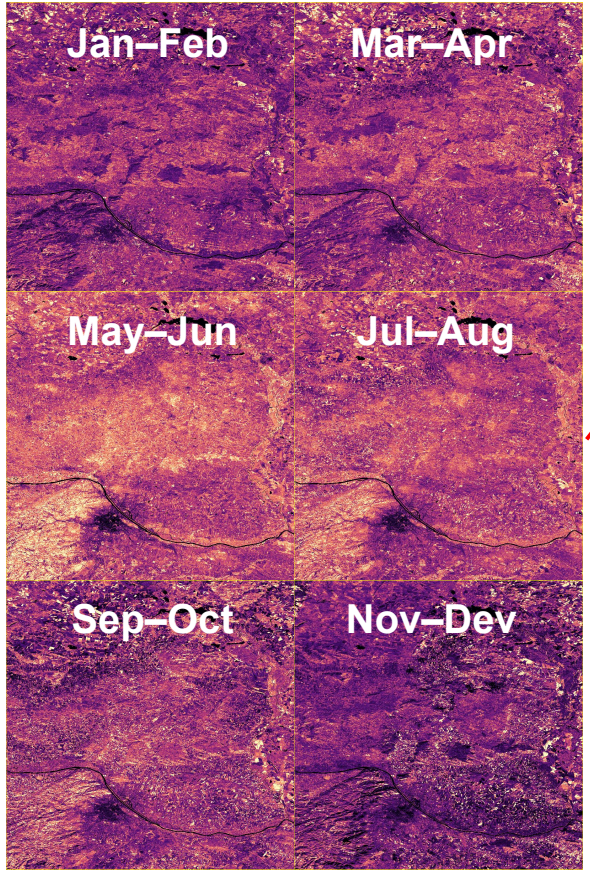
OpenGeoHUB
Connect • Create • Share • Repeat

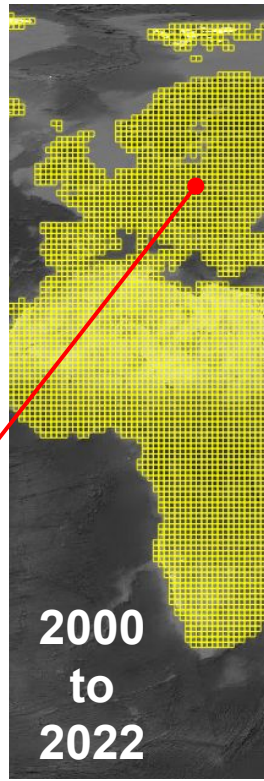
23 years x (7 indices + 7 bands) x 6 dates x 16M pixels

NDVI
EVI
NDWI
NIRv
FAPAR
NDTI
BSI

+

BLUE
GREEN
RED
NIR
SWIR 1
SWIR 2
THERMAL

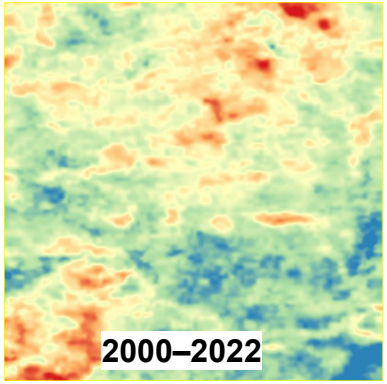




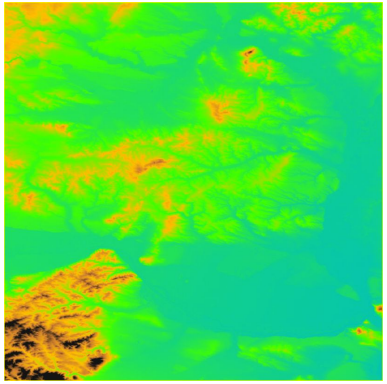
4004 x 4004 pixels

Static layers - MODIS variable, terrain, distance maps

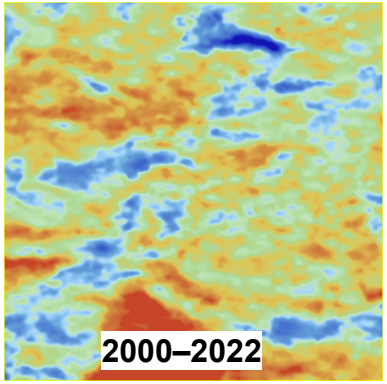
85 static layers x 16M pixels



24 long-term water vapour *(p50 & std)*

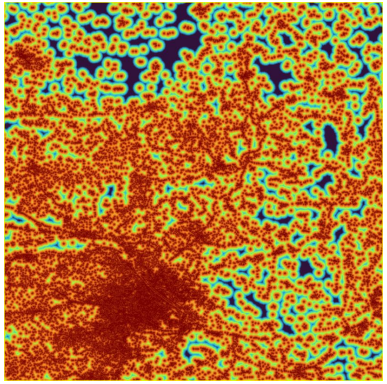


2 DTM static layers



24 long-term daytime temperature *(p50 & std)*

24 long-term nighttime temperature *(p50 & std)*

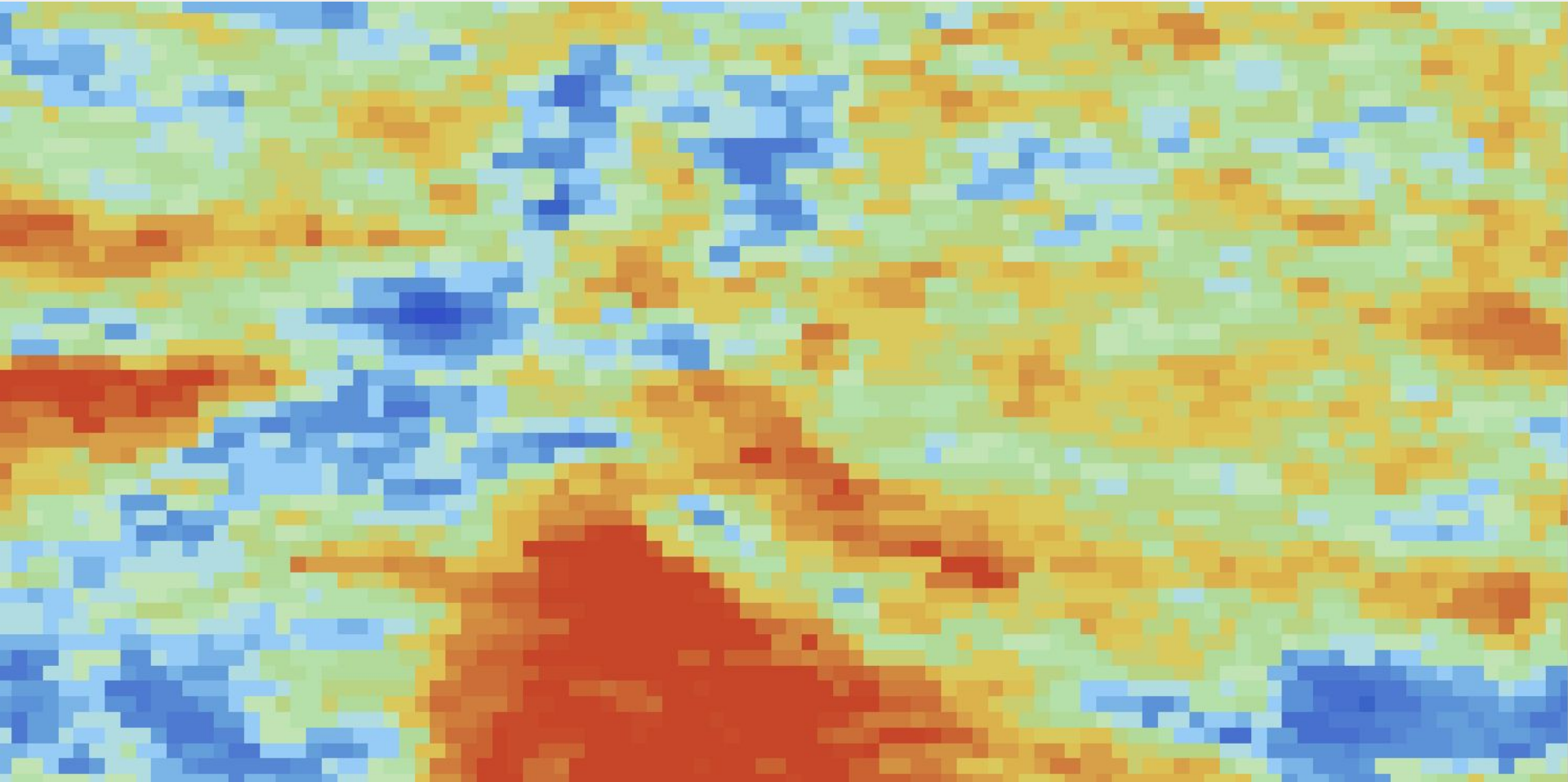


11 Static distance maps

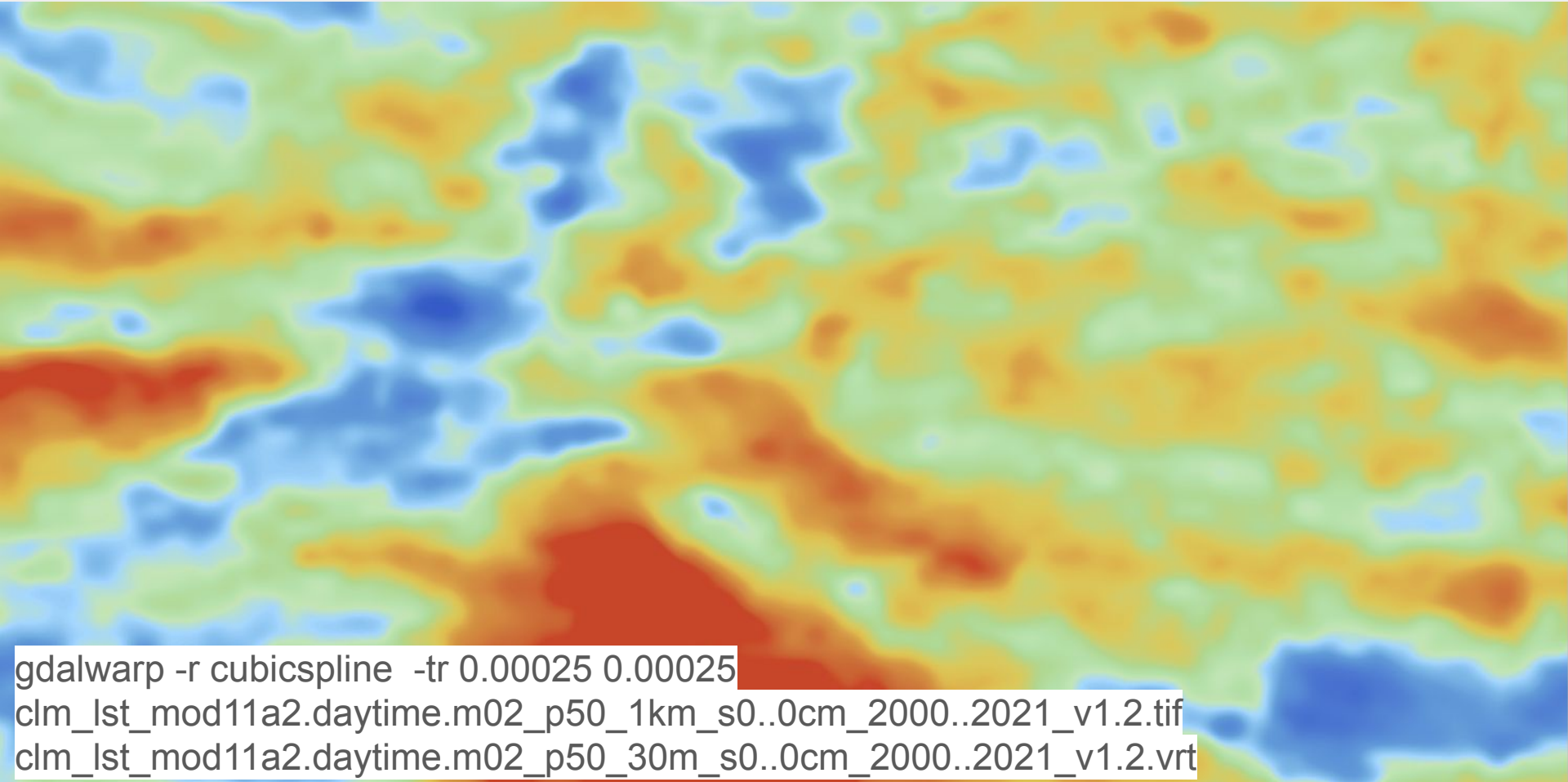
4004 x 4004 pixels

Downscale Trick (1km -> 30m)

OpenGeoHUB
Connect • Create • Share • Repeat



Downscale Trick (1km -> 30m)



Reading all layers & years

OpenGeoHUB
Connect • Create • Share • Repeat

88 static layers — *6.34 secs*

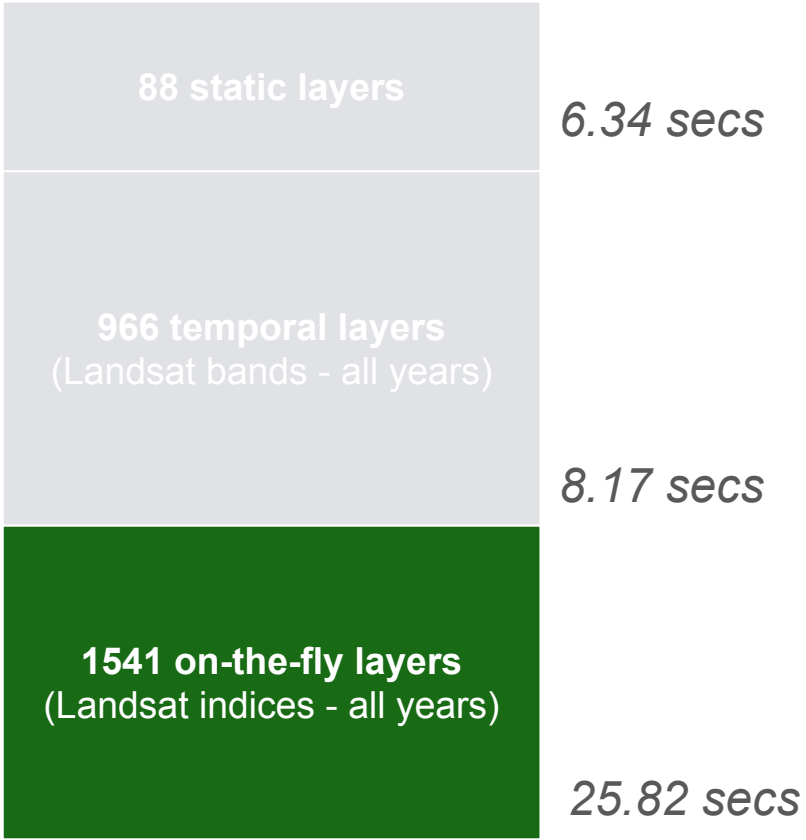
966 temporal layers
(Landsat bands - all years) — *8.17 secs*

1541 on-the-fly layers
(Landsat indices - all years) — *25.82 secs*

(2595, 16032016)

70 Gb RAM

On-the-fly layer calculations



6.34 secs

8.17 secs

25.82 secs

On-the-fly layer calculations

Python libraries to optimize Numpy Computation

Bottleneck

https://bottleneck.readthedocs.io

NumExpr https://numexpr.readthedocs.io

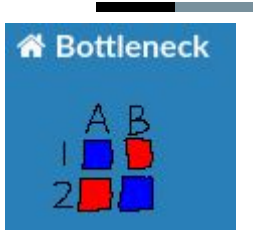
Numba https://numba.pydata.org

High performance computing in Python

colab GitLab

Computational notebooks

On-the-fly layer calculations



Numba

NumExpr

drop-in replacement

```
import bottleneck as bn
bn.nanmedian(red_data, axis=(2))
```

```
from numba import prange, jit

last_dim = red_data.shape[-1]
out_shape = red_data.shape[:-1]

@jit(parallel=True)
def _reduce(data):
    x = np.zeros(len(data))
    for i in prange(data.shape[0]):
        x[i] = np.nanmedian(data[i])
        return x
```

```
import numexpr as ne

expression = '( (nir - red) / (nir + red) ) * nir'
params = { 'nir': nir_data, 'red': red_data }

ne.evaluate(expression, params,
optimization='aggressive')
```

15x speed up (array reduction)

	library	average	std
2	Numba	0.130819	0.002264
1	Bootleneck	0.367630	0.004450
0	Numpy	2.060645	0.203988

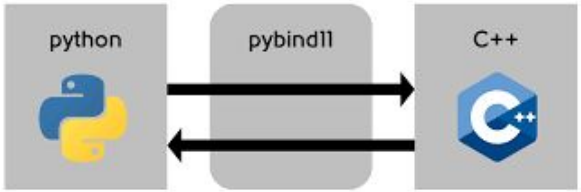
4.5x speed up (mathematical operation)

	library	average	std
1	Numexpr	0.020733	0.000450
2	Numba	0.043981	0.000305
0	Numpy	0.092036	0.002885

On-the-fly layer calculations


```
band_scaling = 0.004
result_scaling = 125.
result_offset = 125.

skmap_bindings.computeNormalizedDifference(data, n_threads,
      swir1_idx, swir2_idx, ndti_idx,
      band_scaling, band_scaling,
      result_scaling, result_offset, [0., 250.])
```

~6x speed up



Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.

```
void TransArray::computeNormalizedDifference(std::vector<uint_t> positive_indices,
                                             std::vector<uint_t> negative_indices,
                                             std::vector<uint_t> result_indices,
                                             float_t positive_scaling,
                                             float_t negative_scaling,
                                             float_t result_scaling,
                                             float_t result_offset,
                                             std::vector<float_t> clip_value)
{
    skmapAssertIfTrue((positive_indices.size() != negative_indices.size()) || (positive_indices.size() != result_indices.size()),
                      "scikit-map ERROR 3: positive_i, negative_i, result_i must be of the same size");
    auto computeNormalizedDifferenceRow = [&] (uint_t i, Eigen::Ref<MatFloat::RowXpr> row)
    {
        uint_t positive_i = positive_indices[i];
        uint_t negative_i = negative_indices[i];
        row = (m_data.row(positive_i) * positive_scaling - m_data.row(negative_i) * negative_scaling).array() /
              (m_data.row(positive_i) * positive_scaling + m_data.row(negative_i) * negative_scaling).array() *
              result_scaling + result_offset;
        row = (row.array()).round();
        row = (row.array() == -inf_v).select(-result_scaling + result_offset, row);
        row = (row.array() == +inf_v).select(result_scaling + result_offset, row);
        row = (row.array() < clip_value[0]).select(clip_value[0], row);
        row = (row.array() > clip_value[1]).select(clip_value[1], row);
    };
    this->parRowPerm(computeNormalizedDifferenceRow, result_indices);
}
```

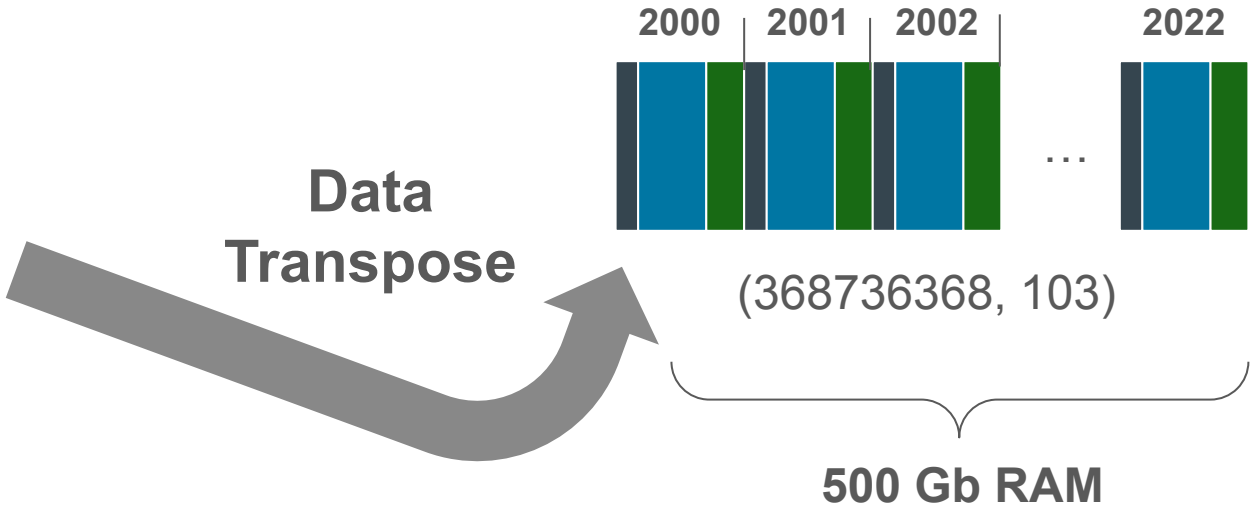
Data transposition

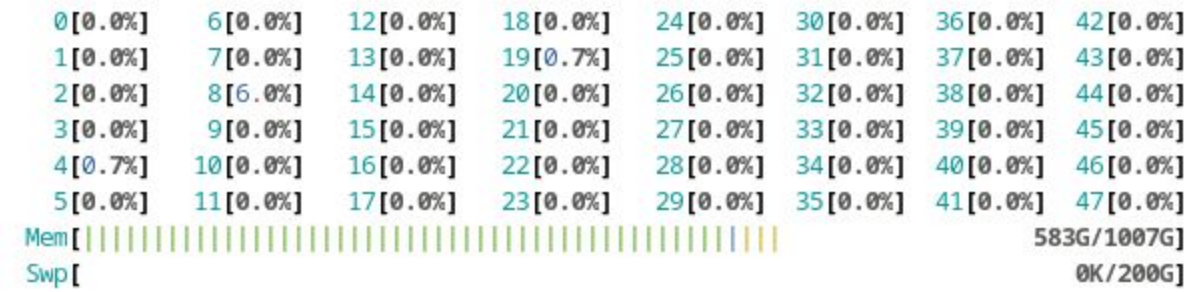
88 static layers

966 temporal layers
(Landsat bands - all years)

1541 on-the-fly layers
(Landsat indices - all years)

(2595, 16032016)





Data transposition

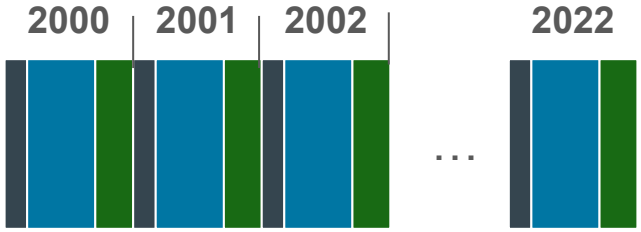
88 static layers

966 temporal layers
(Landsat bands - all years)

1541 on-the-fly layers
(Landsat indices - all years)

(2595, 16032016)

Data Transpose

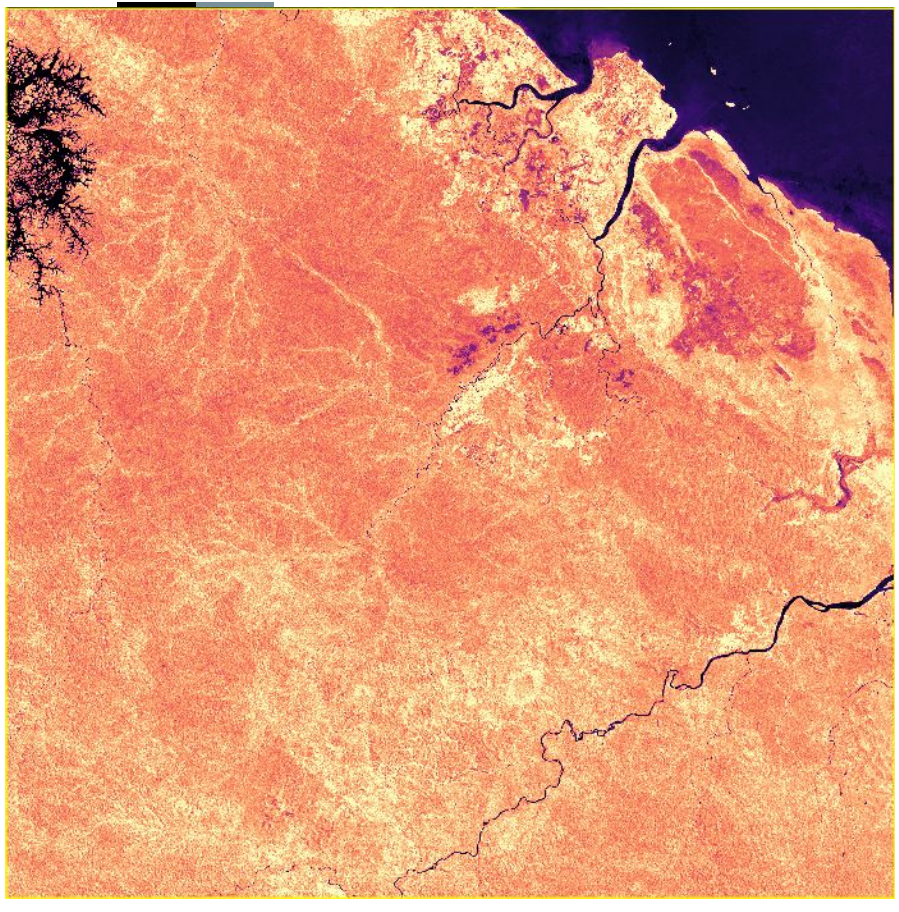


(368736368, 103)

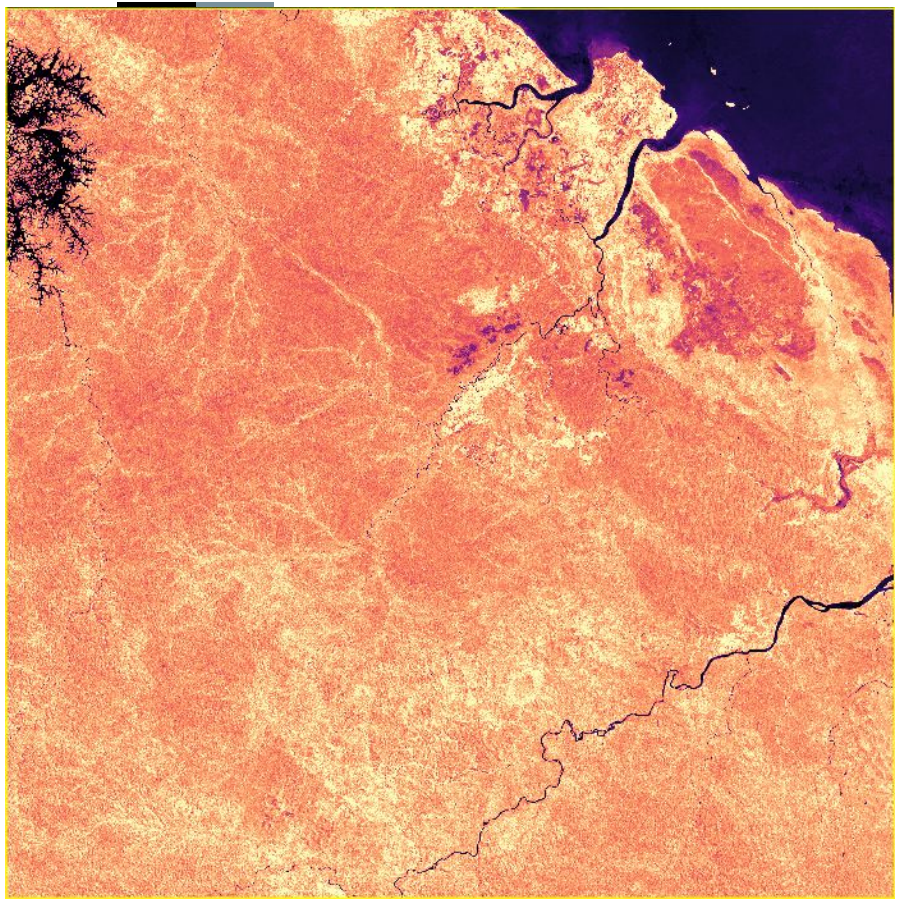
÷ 23

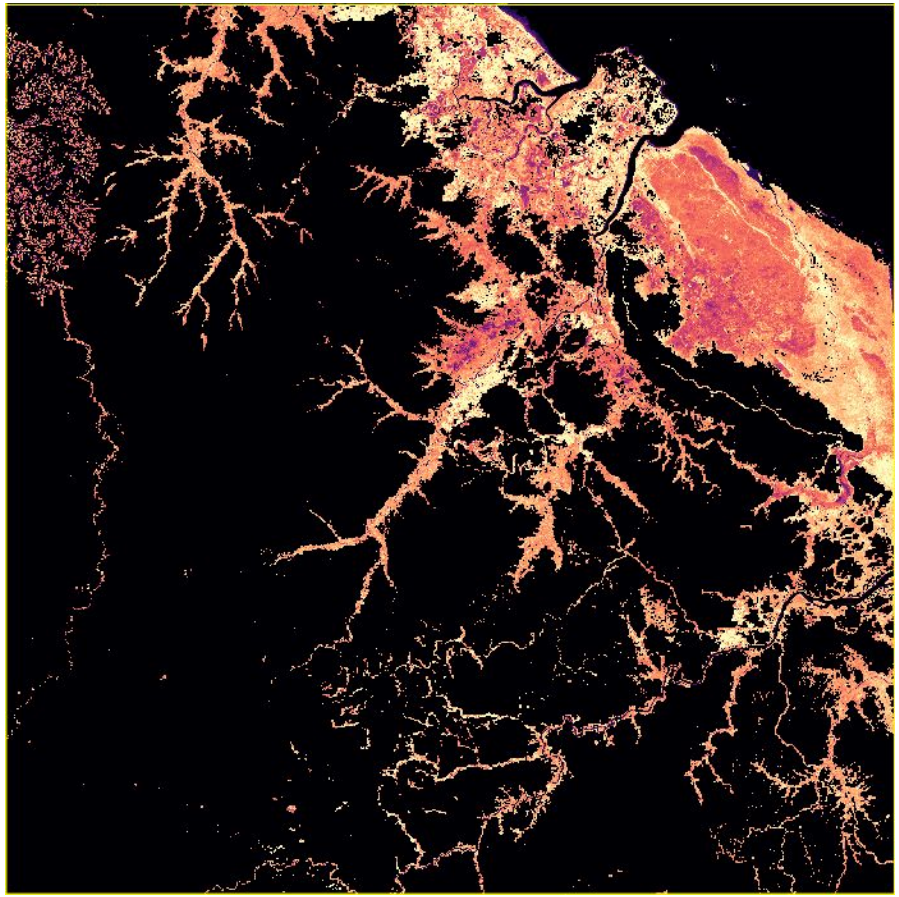
(16032016, 103)

Global prediction mask



Global prediction mask



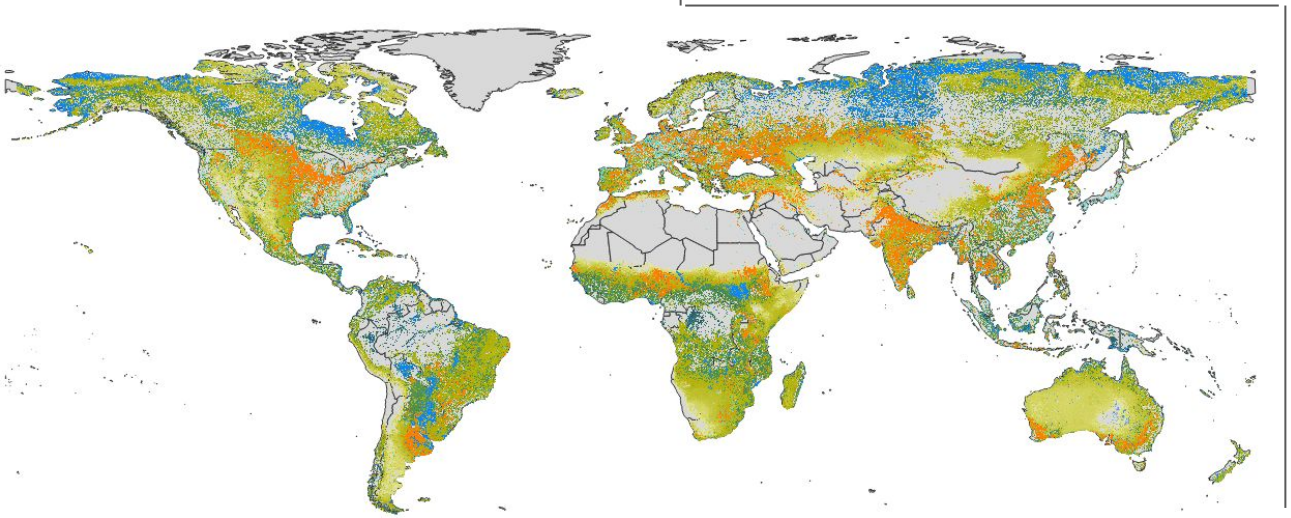


Global prediction mask

Based on [UMD GLAD annual land cover and land use change]

Excluded classes they appear in all maps (2000 –2020)

- Terra Firma, true desert - 3% short veg. cover (0)
- Terra Firma, true desert - 7% short veg. cover (1)
- Terra Firma, semi-arid - 11% short veg. cover (2)
- Terra Firma, semi-arid - 15% short veg. cover (3)
- Terra Firma, semi-arid - 19% short veg. cover (4)
- Terra Firma, semi-arid - 23% short veg. cover (5)



- Wetland, salt pan - 3% short veg. cover (100)
- Wetland, salt pan - 7% short veg. cover (101)
- Terra Firma, stable tree cover - 10m trees (32)
- Terra Firma, stable tree cover - 11m trees (33)
- Terra Firma, stable tree cover - 12m trees (34)
- Terra Firma, stable tree cover - 13m trees (35)
- Terra Firma, stable tree cover - 14m trees (36)
- Terra Firma, stable tree cover - 15m trees (37)
- Terra Firma, stable tree cover - 16m trees (38)
- Terra Firma, stable tree cover - 17m trees (39)
- Terra Firma, stable tree cover - 18m trees (40)
- Terra Firma, stable tree cover - 19m trees (41)
- Terra Firma, stable tree cover - 20m trees (42)
- Terra Firma, stable tree cover - 21m trees (43)
- Terra Firma, stable tree cover - 22m trees (44)
- Terra Firma, stable tree cover - 23m trees (45)
- Terra Firma, stable tree cover - 24m trees (46)
- Terra Firma, stable tree cover - 25m trees (47)
- Terra Firma, stable tree cover - >25m trees (48)
- Snow/ice, stable (241)
- Ocean (254)
- No data (255)

Global prediction mask

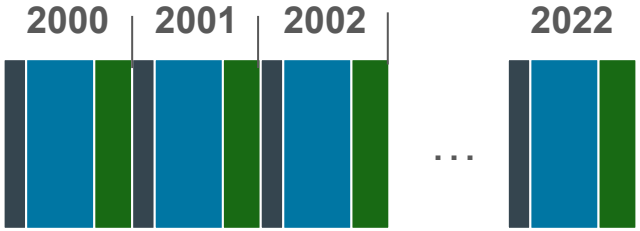
88 static layers

966 temporal layers
(Landsat bands - all years)

1541 on-the-fly layers
(Landsat indices - all years)

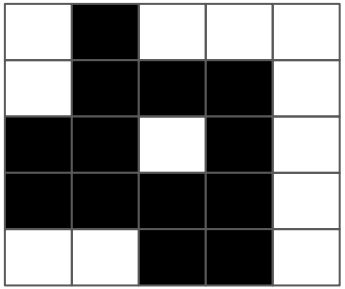
(2595, 16032016)

Data Transpose

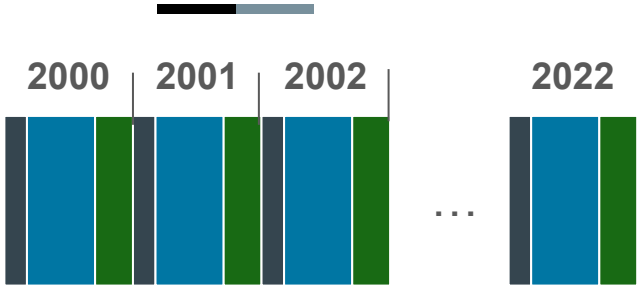


(309995587, 103)

Apply prediction mask



(Finally) **Global ML prediction**



(309995587, 103)

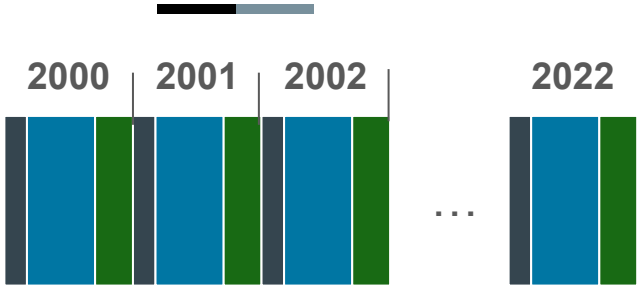
Same input features

ML model 1
(cultivated grassland RF)

ML model 2
(Natural/semi-grassland RF)

```
output = model.predict(data)
# OR
output = model.predict_proba(data)
```

(Finally) **Global ML prediction**



(309995587, 103)

Same input features

ML model 1
(cultivated grassland RF)

ML model 2
(Natural/semi-grassland RF)

scikit learn

DMLC XGBoost

CatBoost

MAPIE

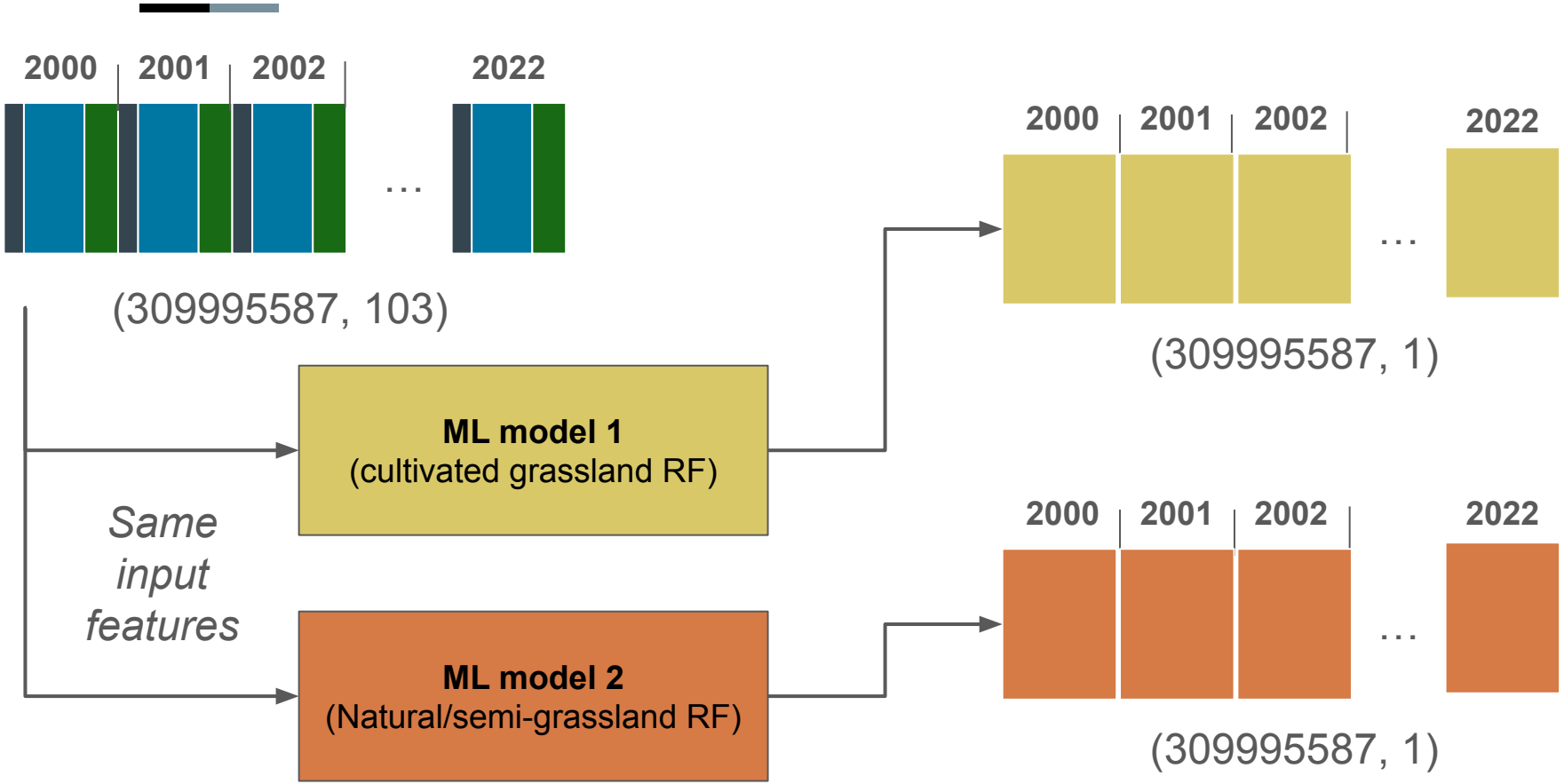
SKORCH

SciKeras

https://github.com/krzjoa/awesome-python-data-science

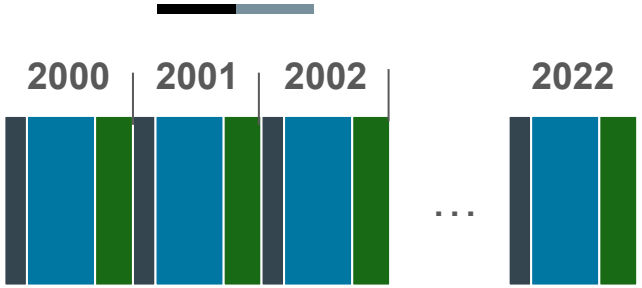
```
output = model.predict(data)
# OR
output = model.predict_proba(data)
```

(Finally) **Global ML prediction**



(Finally) **Global ML prediction**

Runtime for each sklearn model: **800 secs**



(309995587, 103)

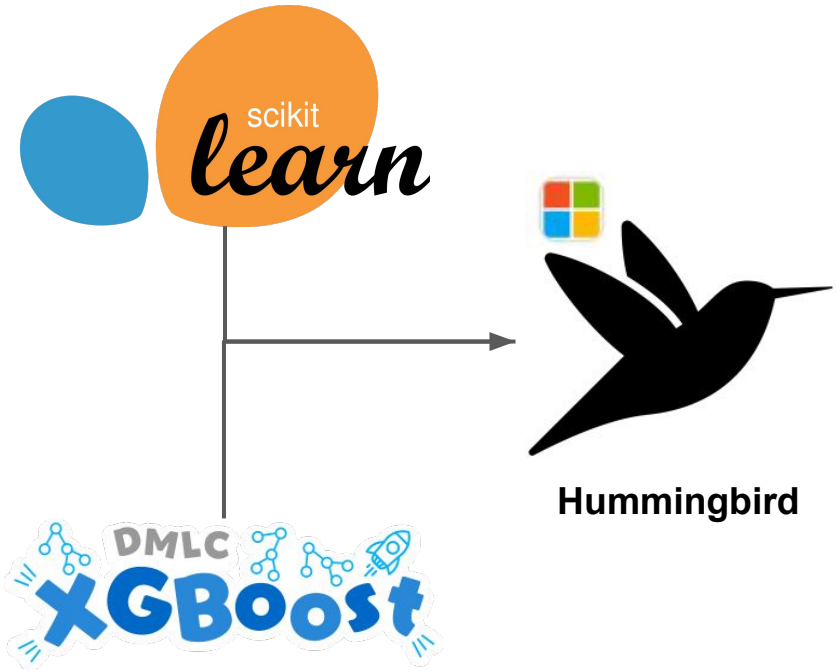
Same input features

ML model 1
(cultivated grassland RF)

ML model 2
(Natural/semi-grassland RF)

(309995587, 1)

Compiling ML models



Hummingbird is a library for compiling trained traditional ML models into tensor computations.

https://github.com/microsoft/hummingbird

Compiling ML models

ONNX

Open standard for machine learning interoperability

Hummingbird

Deep learning compiler that enables access to
high-performance machine learning anywhere for everyone

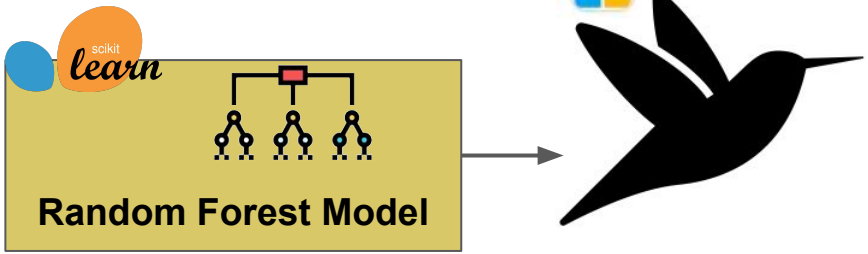
PyTorch & TorchScript

Hummingbird is a library for compiling trained traditional ML models into tensor computations.

https://github.com/microsoft/hummingbird

TorchScript is a way to create serializable and optimizable models from PyTorch code

Compiling ML models



Hummingbird

Suitable for CPU inference

tvm

Deep learning compiler that enables access to high-performance machine learning anywhere for everyone

Hummingbird is a library for compiling trained traditional ML models into tensor computations.

https://github.com/microsoft/hummingbird

Compiling ML models

OpenGeoHUB
Connect • Create • Share • Repeat

scikit learn

Random Forest Model

Hummingbird

tvm

Deep learning compiler that enables access to
rmance machine learning anywhere for everyone

Suitable for CPU inference

Hummingbird is a library for compiling trained traditional ML models into tensor computations.

https://github.com/microsoft/hummingbird

Compiling ML models

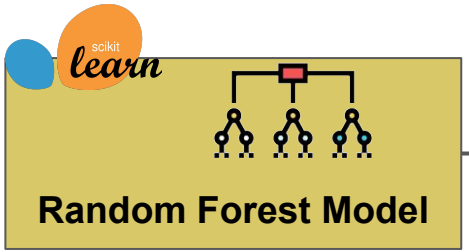
ML model 1
(cultivated grassland RF)

Number of trees: **60**
Max. depth (avg): **33.7**
Number of nodes (avg): **13,716**

ML model 2
(Natural/semi-grassland RF)

Number of trees: **60**
Max. depth (avg): **40.7**
Number of nodes (avg): **24,029**

Compiling ML models




```c
#include "header.h"
void predict_margin_multiclass_unit10(union Entry* data, double* result) {
  double sum[3] = {0};
  unsigned int tmp;
  int nid, cond, fid;  /* used for folded subtrees */
  if ( UNLIKELY( !(data[3].missing != -1) || (data[3].qvalue <= 24) ) ) {
    if ( LIKELY( !(data[5].missing != -1) || (data[5].qvalue <= 146) ) ) {
      if ( LIKELY( !(data[71].missing != -1) || (data[71].qvalue <= 652) ) ) {
        if ( UNLIKELY( !(data[84].missing != -1) || (data[84].qvalue <= 224) ) ) {
          if ( LIKELY( !(data[63].missing != -1) || (data[63].qvalue <= 678) ) ) {
            if ( LIKELY( !(data[49].missing != -1) || (data[49].qvalue <= 8718) ) ) {
              if ( UNLIKELY( !(data[77].missing != -1) || (data[77].qvalue <= 448) ) ) {
                if ( UNLIKELY( !(data[93].missing != -1) || (data[93].qvalue <= 594) ) ) {
                  if ( LIKELY( !(data[25].missing != -1) || (data[25].qvalue <= 1050) ) ) {
                    if ( UNLIKELY( !(data[29].missing != -1) || (data[29].qvalue <= 2776) ) ) {
                      sum[0] += (double)0;
                      sum[1] += (double)0;
                      sum[2] += (double)1;
                    } else {
                      if ( LIKELY( !(data[33].missing != -1) || (data[33].qvalue <= 726) ) ) {
                        sum[0] += (double)0;
                        sum[1] += (double)0.5952380952380952328;
                        sum[2] += (double)0.40476190476190476672;
                      } else {
                        sum[0] += (double)0;
                        sum[1] += (double)0;
                        sum[2] += (double)1;
                      }
                    }
                  } else {
...
```

Model compiler for decision tree models

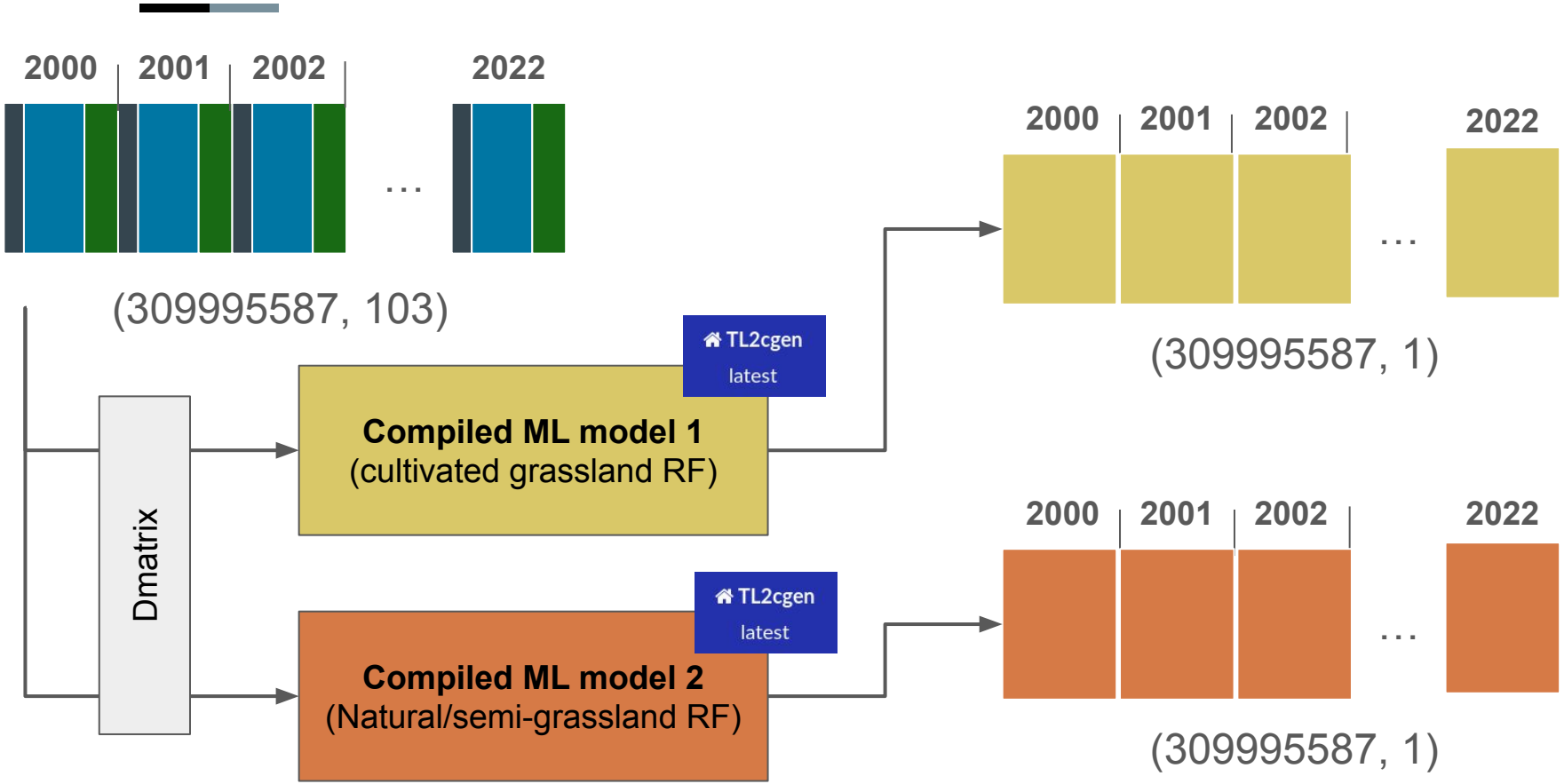
https://tl2cgen.readthedocs.io

C++ compiler

One IF-ELSE per tree node

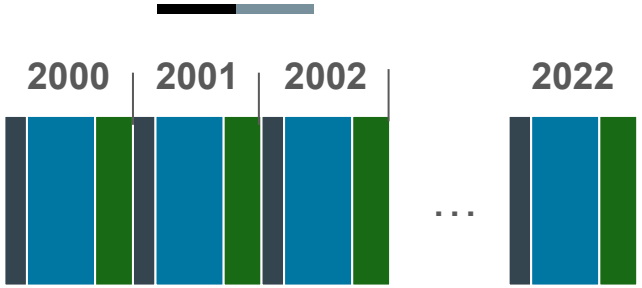
The Performance of Decision Tree Evaluation Strategies

(Finally) **Global ML prediction**



(Finally) **Global ML prediction**

OpenGeoHUB
Connect • Create • Share • Repeat



(309995587, 103)

Creating dmatrix: **78 secs**
Runtime for each compiled model: **28 secs**

8x speed-up

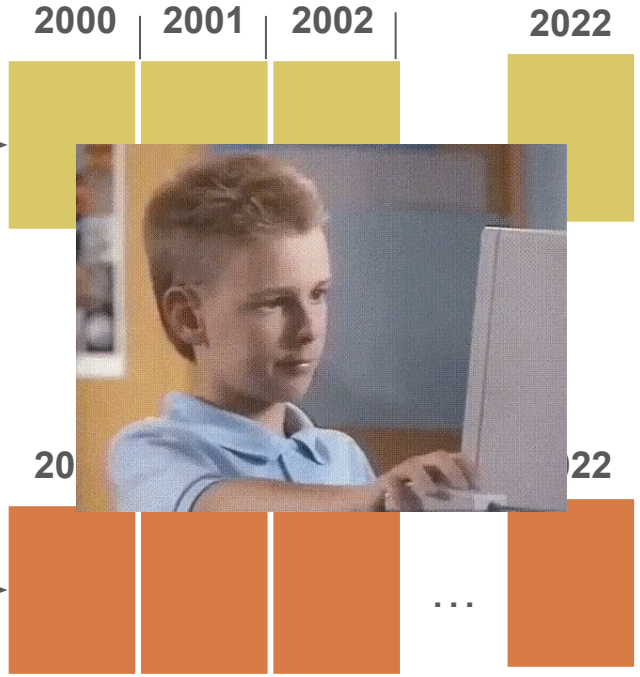
Dmatrix

TL2cgen
latest

Compiled ML model 1
(cultivated grassland RF)

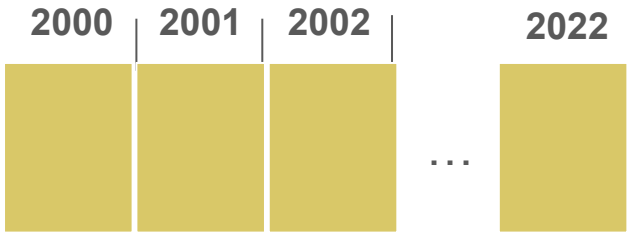
TL2cgen
latest

Compiled ML model 2
(Natural/semi-grassland RF)



(309995587, 1)

(Finally) **Global ML prediction**



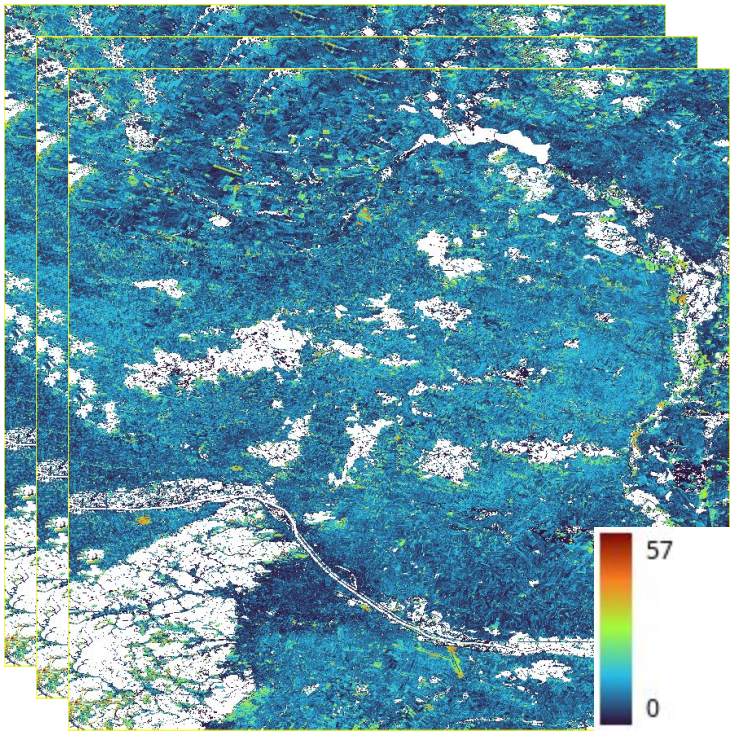
(309995587, 1)

Data "back-transpose"

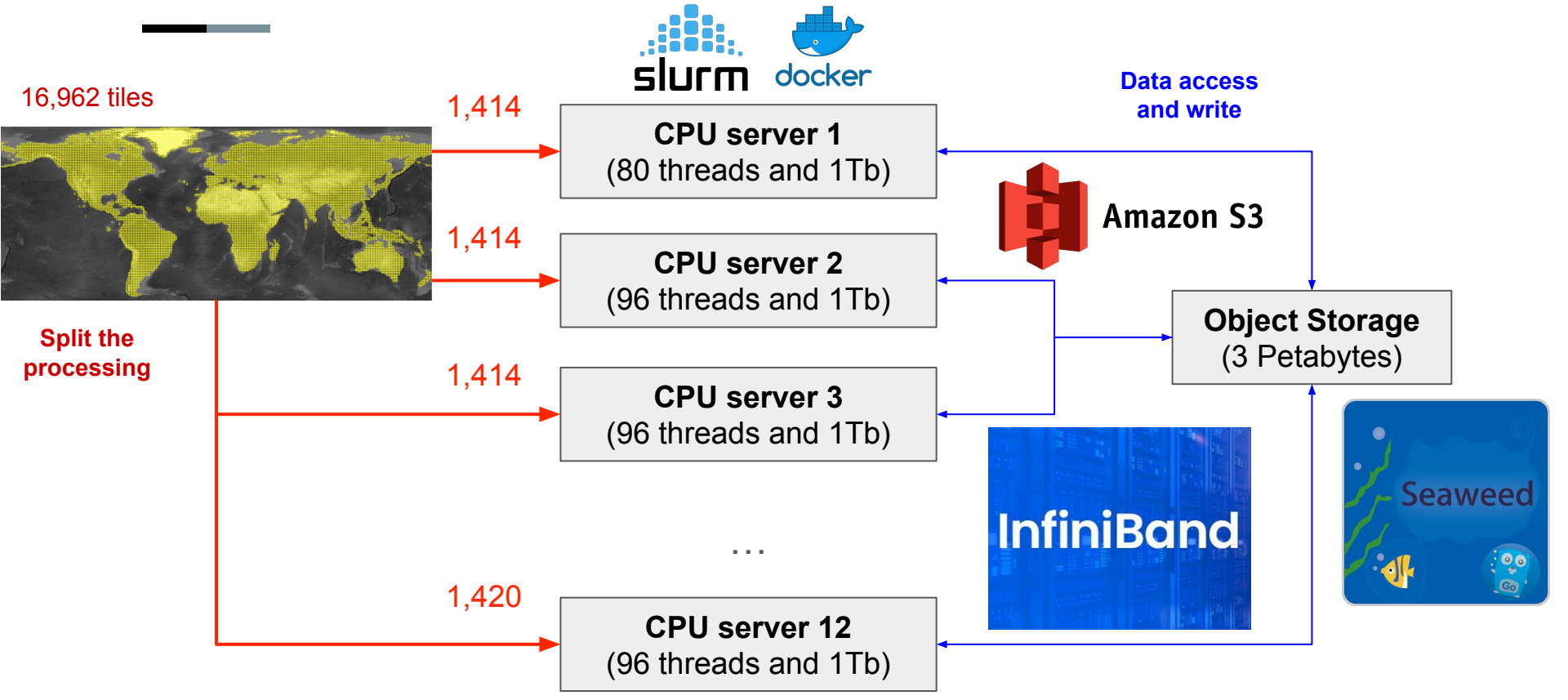
Probability output

(23, 16032016)

Cultivated grassland maps



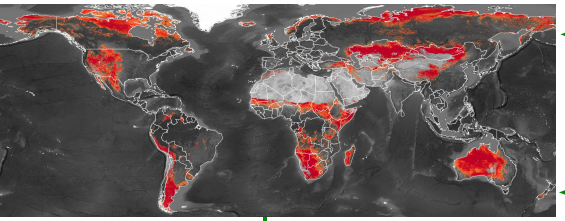
(Finally) **Global ML prediction**



(Finally) **Global ML prediction**

OpenGeoHUB
Connect • Create • Share • Repeat

slurm docker



CPU server 1
(80 threads and 1Tb)

CPU server 2
(96 threads and 1Tb)

CPU server 3
(96 threads and 1Tb)

...

CPU server 12
(96 threads and 1Tb)

Data access

Amazon S3

Object Storage
(3 Petabytes)

InfiniBand

Seaweed

GDAL

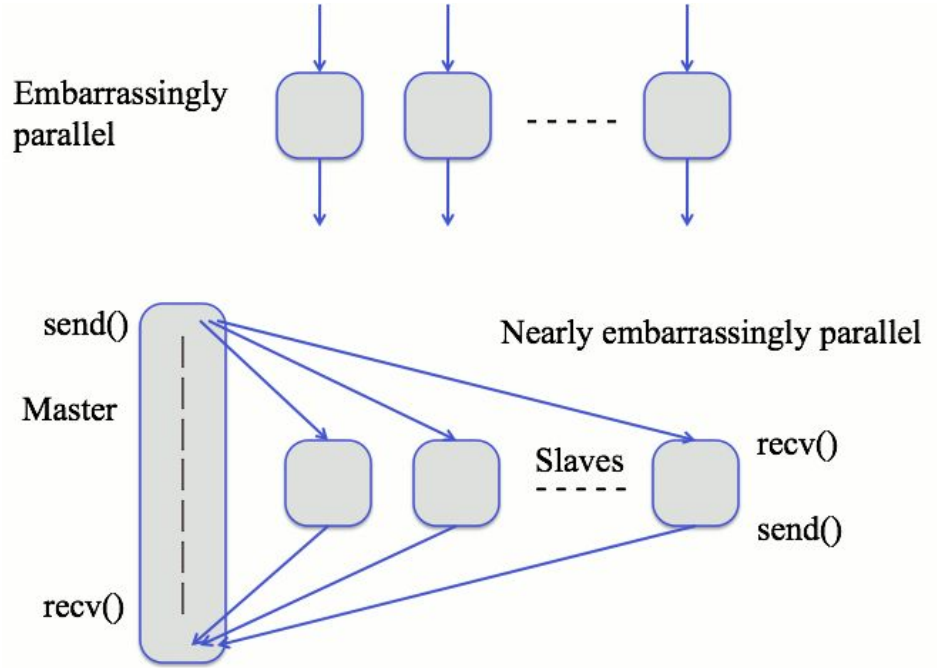
Cloud Optimized

GeoTIFF generation

gdalbuildvrt
gdal_translate

Embarrassingly parallel problems

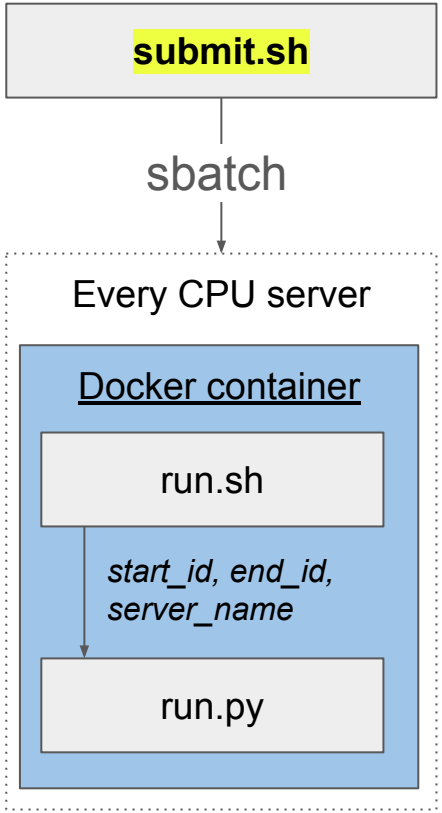
- Can be divided into completely **independent** parts,
- Requires none or very little communication,
- **Nearly embarrassingly parallel** is an embarrassingly parallel computation that requires initial data to be distributed and final results to be collected in some way



Source: Embarrassingly Parallel Computations & Embarrassingly Parallel Algorithms

Implementation in slurm

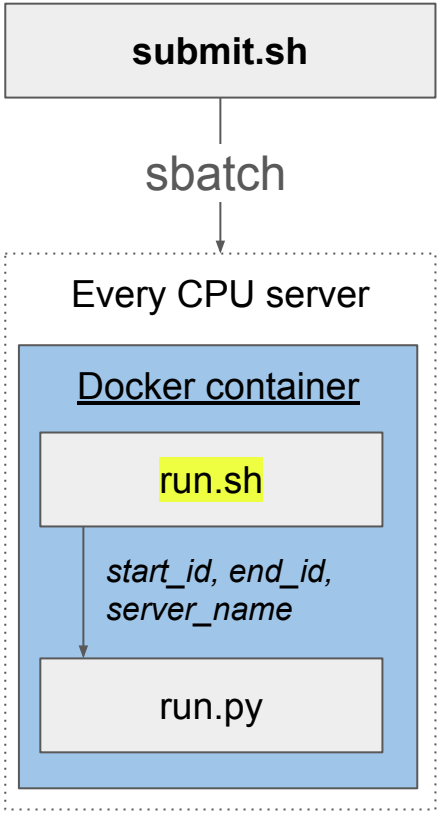
```
#!/bin/bash
#SBATCH --partition=cpu                                  # Partition name (cpu, gpu, nas, all)
#SBATCH --ntasks-per-node=1                              # Do not change it
#SBATCH --ntasks=1                                       # Do not change it
#SBATCH --job-name=wri_pasture_class                     # Job name
#SBATCH --array=1-12                                     # Number of server to use
#SBATCH --mail-type=ALL                                  # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=leandro.parente@opengeohub.org       # Where to send mail
#SBATCH --output=%x_%N.log                               # Do not change it
#SBATCH --exclude=landmark,bender  # Servers to exclude (landmark,primus)

# Slurm helper functions
source /mnt/slurm/jobs/ogh_slurm.sh

# Total number of ids to split and process across the servers
N_IDS=16962 # FAPAR

# Python docker container
DOCKER_IMAGE=192.168.49.30:5000/pygeo-ide:v3.8.16-mkl-gdal362-pasture_class
LANG=Python

execute
```



Implementation in slurm

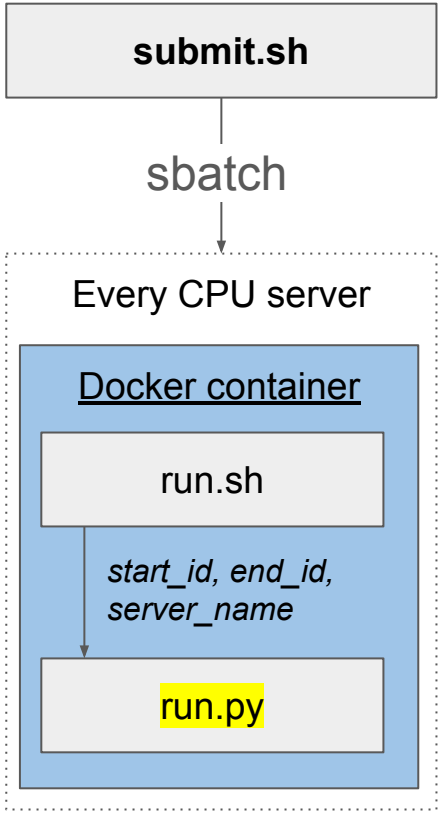
OpenGeoHUB
Connect • Create • Share • Repeat

```
#!/bin/bash

job_command=( "$@" )

######################################################
# Intall all necessary libraries inside the container
######################################################
pip install -e 'git+https://github.com/openlandmap/scikit-map.git@pasture_class_prod#egg=scikit-map[full]'
pip install -e 'git+https://github.com/leiferlab/savitzkygolay#egg=savitzkygolay'

mc alias set g1 http://192.168.49.30:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g2 http://192.168.49.31:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g3 http://192.168.49.32:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g4 http://192.168.49.33:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g5 http://192.168.49.34:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g6 http://192.168.49.35:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g7 http://192.168.49.36:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g8 http://192.168.49.37:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g9 http://192.168.49.38:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g10 http://192.168.49.39:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g11 http://192.168.49.40:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g12 http://192.168.49.41:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4
mc alias set g13 http://192.168.49.42:8333 iwum9G1fEQ920lYV4ol9 GMBME3Wsm8S7mBXw3U4CNWurkzWMqGZ0n2rXHggS0 --api S3v4

cd /mnt/slurm/jobs/wri_pasture_class/
mkdir /mnt/$(hostname)/wri_pasture_class

cp -Rv /mnt/slurm/jobs/wri_pasture_class/models/compiled /mnt/$(hostname)/wri_pasture_class/

echo "######################################################"
echo $job_command
echo "######################################################"
echo "Starting: "$(date)
$job_command
echo "Ending: "$(date)
```



Implementation in slurm

```python
import time
import sys

start_id=int(sys.argv[1])
end_id=int(sys.argv[2])
server_name=sys.argv[3]

print(f"Python: Processing the ids {start_id}-{end_id} in {server_name}")
time.sleep(10)
print(f"Python: ...actually {server_name} was just chilling out")
```



Main takeaways

- The current ML prediction pipeline is complex and difficult to maintain, however reduced the production time from **40 to 7 days**,
- We are encapsulating C++ functions into scikit-map library (https://github.com/openlandmap/scikit-map),

- We observed huge improvement in performance after migrate from **Python multiprocessing** to **C++ multithreading** (more information about the Global Interpreter Lock GIL),
- Most of the users **do not need** such complex pipeline, however part of the optimizations presented might be useful for other applications (ex: parallel reading, on-the-fly calculation, third-party libraries).
- The full reproducibility of this pipeline depends on have the **input EO data** publicly accessible, **container with all libraries** used and **better code abstraction** for the optimizations implemented.

Leandro Parente

leandro.parente@opengeohub.org

https://opengeohub.org

OpenGeoHUB

Connect • Create • Share • Repeat