# An introduction to Google Earth Engine

*Exploring geospatial data for Machine Learning*

EuroSciPy 2023

Duarte O.Carmo

duarteocarmo.com - @duarteocarmo

# Hello! I’m Duarte.

*/du-art/* - it’s Portuguese

ML/Software Engineer & contractor

From Portugal, based in Copenhagen, Denmark

I like **running**, and **writing** on my blog

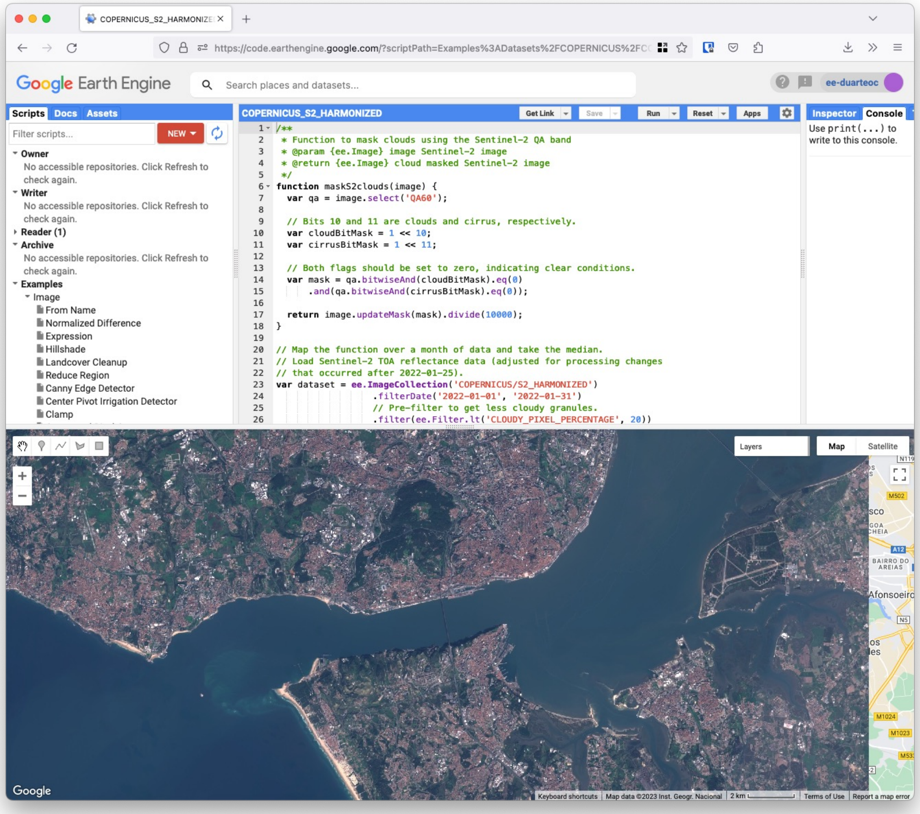
**Past**: Strategy, Product Mgmt., New Ventures, Mgmt. Consulting

**Now**: I help companies solve **tough** problems end-to-end

# Today's talk will give you a glimpse into the possibilities of Geospatial data

- Geospatial data and why it matters
- Google Earth Engine
- Example and Machine Learning Applications

- I'm not a "Geospatial Engineer"
- I'm not selling you things
- Disagree? Great – raise your hand!



# What do you mean by Geospatial?

# Geospatial data: All data generated from observing our planet in space

**Images:** RGB bands from the Sentinel-2

**Temperature:** From multi/hyper spectral satellites

**Vegetation**: NDVI (vegetation index)

**Landcover:** Is this a city? Farm?

**Precipitation**: How much rainfall?

...

# Geospatial data is one of the richest sources of information…

Hundreds of satellites orbiting earth

Multiple providers (NASA, EU, etc.)

Amazing derived datasets

Free! (A large portion of it)

## Geospatial data is one of the richest sources of information...

Hundreds of satellites orbiting earth

Multiple providers (NASA, EU, etc.)

Amazing derived datasets

Free! (A large portion of it)

## ... but handling and accessing it can be a real pain

Every satellite is different

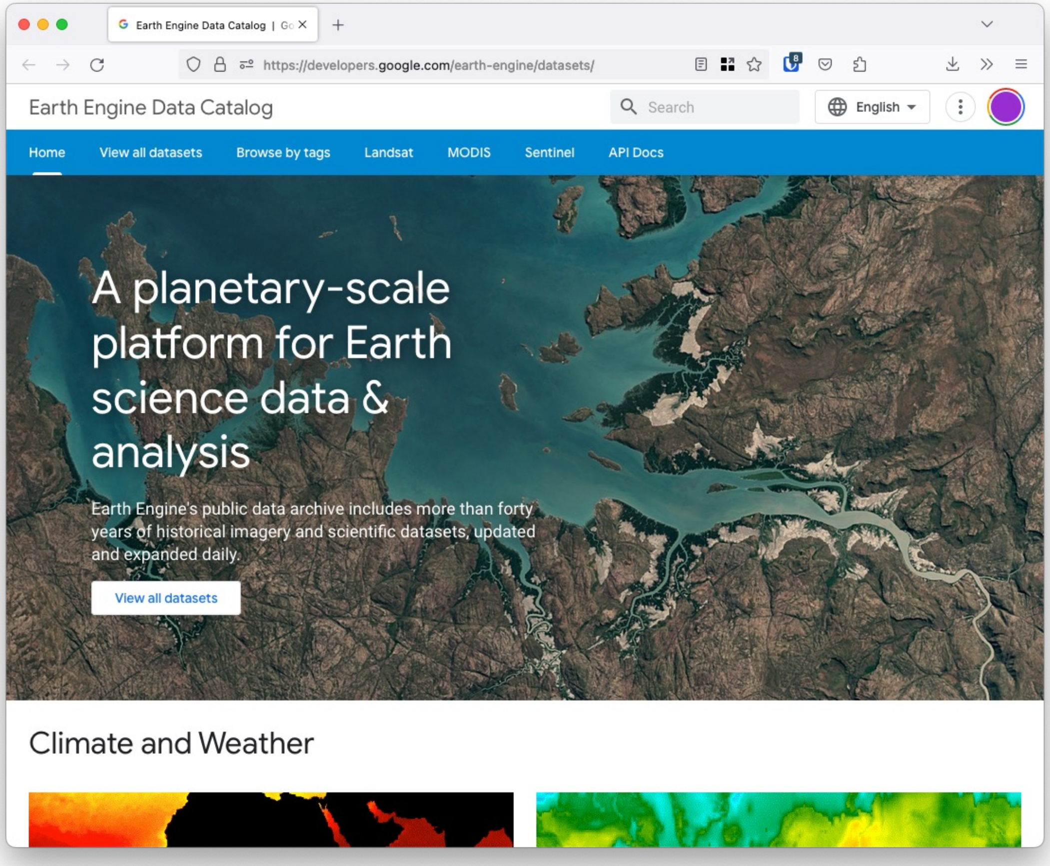
Every provider has a different portal

Different units, different resolutions

Potential data engineering **hell!!**

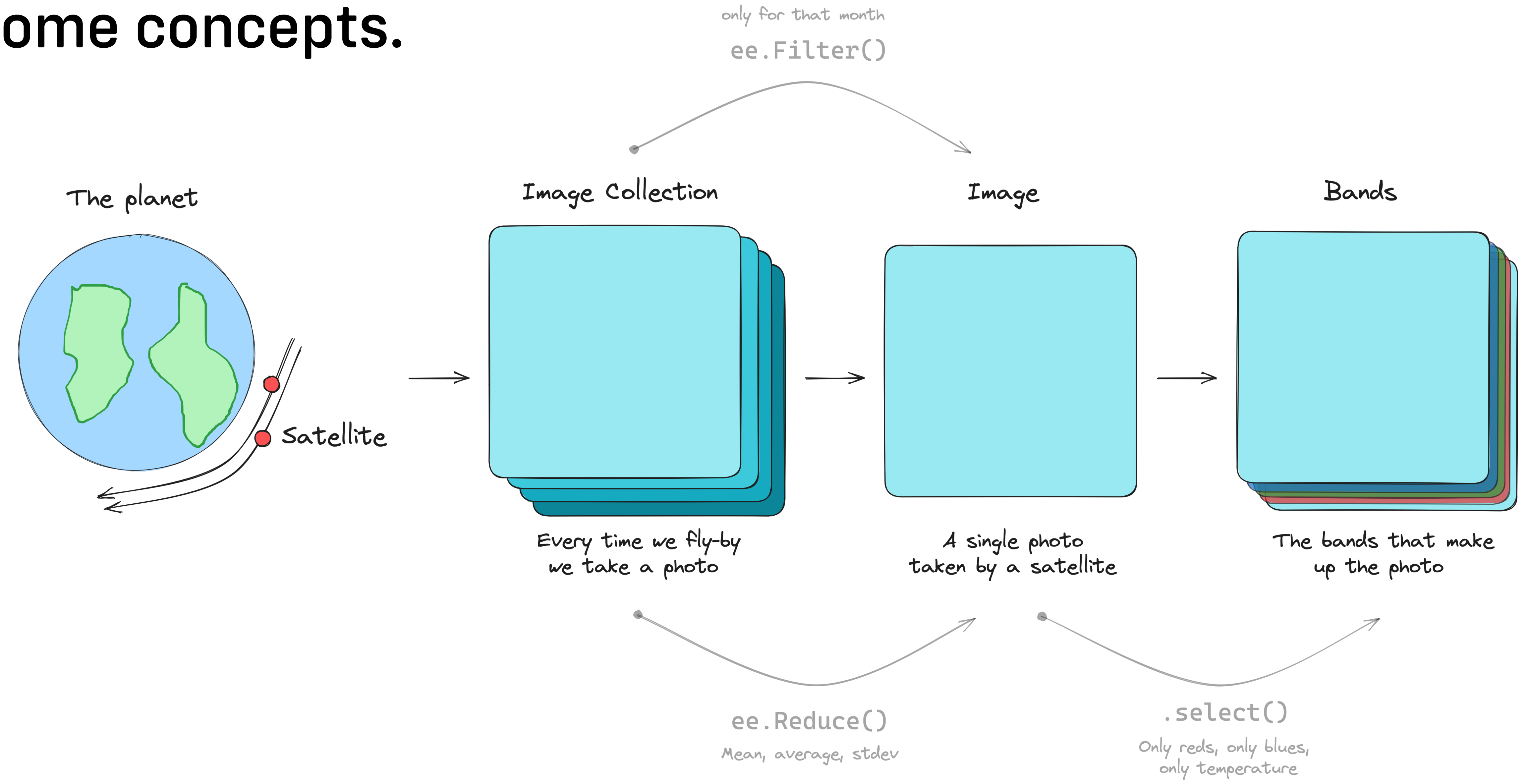
# Introducing: Google Earth Engine

# GEE solves most pains when accessing and manipulating Geospatial data

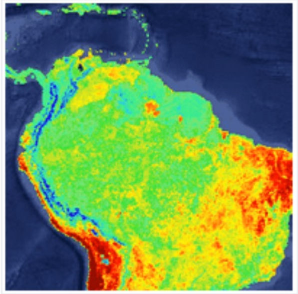



- One API (JavaScript and Python)
- Unified docs for all products
- Compute platform (joins, merges, filters)
- Free for research purposes
- Continuously updating catalog

# Some concepts.



# MOD11A2.061 Terra Land Surface Temperature and Emissivity 8-Day Global 1km



**Dataset Availability**

2000-02-18T00:00:00Z–2023-07-04T00:00:00

For when is this data available?

**Dataset Provider**

NASA LP DAAC at the USGS EROS Center

**Earth Engine Snippet**

`ee.ImageCollection("MODIS/061/MOD11A2")`

Image Collection or Image?

**Tags**

8-day emissivity global lst mod11a2 modis nasa surface-temperature terra usgs

Description of all bands

Description | Bands | Terms of Use | Citations | DOIs

**Resolution**
1000 meters

What is the resolution?

**Bands**

Band details

| Name | Units | Min | Max | Scale | Offset | Description |
|---|---|---|---|---|---|---|
| `LST_Day_1km` | K | 7500 | 65535 | 0.02 | | Day land surface temperature |
| `QC_Day` | | | | | | Daytime LST quality indicators |
| Bitmask for QC_Day | | | | | | |
| `Day_view_time` | h | 0 | 240 | 0.1 | | Local time of day observation |
| `Day_view_angl` | deg | 0 | 130 | | -65 | View zenith angle of day observation |
| `LST_Night_1km` | K | 7500 | 65635 | 0.02 | | Night land surface temperature |
| `QC_Night` | | | | | | Nighttime LST quality indicators |
| Bitmask for QC_Night | | | | | | |
| `Night_view_time` | h | 0 | 240 | 0.1 | | Local time of night observation |

## Example

# Mean day time temperature

```python
import ee

# EuroScipy
latitude = 47.558508
longitude = 7.583762
point = ee.Geometry.Point([longitude, latitude])

# Get the image collection
image_collection = ee.ImageCollection("MODIS/061/MOD11A2")

# Process the collection
image = (
    image_collection.filterDate("2023-01-01", "2023-08-01")  # filter by date
    .select(["LST_Day_1km", "LST_Night_1km"])  # select bands
    .mean()  # get the average
    .multiply(0.02)  # adjust for scale
    .subtract(273.15)
)

# Get the results
result = image.sampleRegions(collection=point).getInfo()
result.get("features")[-1].get("properties")
# {'LST_Day_1km': 13.124166666666724, 'LST_Night_1km': 4.775000000000034}
```

# Machine Learning Applications

# Tabular/Timeseries | Using Google Earth Engine to extract rainfall data as a time series

| | country | long | lat | maincause | displaced | date | 0_30d_t0_rainfall_m | 1_30d_t1_rainfall_m | 2_30d_t2_rainfall_m | 3_30d_t3_rainfall_m |
|---|---|---|---|---|---|---|---|---|---|---|
| 4967 | India | 79.226600 | 17.344400 | Monsoonal Rain | 2100 | 2020-10-16 00:00:00 | 0.619084 | 1.037469 | 0.242630 | 0.302843 |
| 4964 | Vietnam | 107.833000 | 15.854800 | Heavy Rain | 900000 | 2020-10-06 00:00:00 | NaN | NaN | NaN | NaN |
| 4963 | Mozambique | 38.127600 | -14.581300 | Heavy Rain | 4000 | 2020-10-02 00:00:00 | NaN | NaN | NaN | NaN |
| 4966 | Togo | 0.777334 | 9.710160 | Heavy Rain | 16000 | 2020-09-15 00:00:00 | 0.099871 | 0.075639 | 0.060188 | 0.513882 |
| 4965 | Nigeria | 6.258020 | 7.774120 | Heavy Rain | 25000 | 2020-09-15 00:00:00 | 4.037949 | 1.330742 | 4.710331 | 3.221587 |
| 4962 | USA | -86.579600 | 32.650800 | Tropical Storm Sally | 0 | 2020-09-15 00:00:00 | 0.017025 | 0.019261 | 0.060149 | 0.044176 |
| 4961 | India | 95.037700 | 27.848600 | Monsoonal Rain | 0 | 2020-09-13 00:00:00 | NaN | NaN | NaN | NaN |
| 4950 | Afghanistan | 67.720900 | 34.916000 | Torrential Rain | 0 | 2020-08-25 00:00:00 | 0.843814 | 0.881901 | 0.289106 | 0.776603 |
| 4951 | Pakistan | 68.215700 | 26.957300 | Monsoonal Rain | 1300 | 2020-08-24 00:00:00 | 0.730026 | 2.174792 | 0.494846 | 0.542660 |
| 4960 | USA | -70.941700 | 19.564200 | Tropical Storm Laura | 600000 | 2020-08-22 00:00:00 | 0.391254 | 0.358152 | 0.596996 | 0.543690 |
| 4954 | Haiti | -70.904600 | 18.907900 | Tropical Storm Laura | 0 | 2020-08-21 00:00:00 | 0.493560 | 0.363616 | 0.596426 | 0.601292 |
| 4959 | India | 77.469800 | 24.709500 | Monsoonal Rain | 60 | 2020-08-21 00:00:00 | 0.195524 | 0.190712 | 0.053476 | 0.149555 |
| 4953 | Kenya | 37.202100 | 2.750120 | Dam Release and Heavy Rain | 5000 | 2020-08-20 00:00:00 | NaN | NaN | NaN | NaN |
| 4955 | Uganda | 30.333400 | 0.385265 | Torrential Rain | 0 | 2020-08-19 00:00:00 | 0.006394 | 0.006034 | 0.010361 | 0.001781 |
| 4952 | Chad | 14.646100 | 10.199000 | Heavy Rain | 38000 | 2020-08-10 00:00:00 | 0.609008 | 0.030559 | 0.008041 | 0.000344 |

Rainfall (30d periods from observation date)

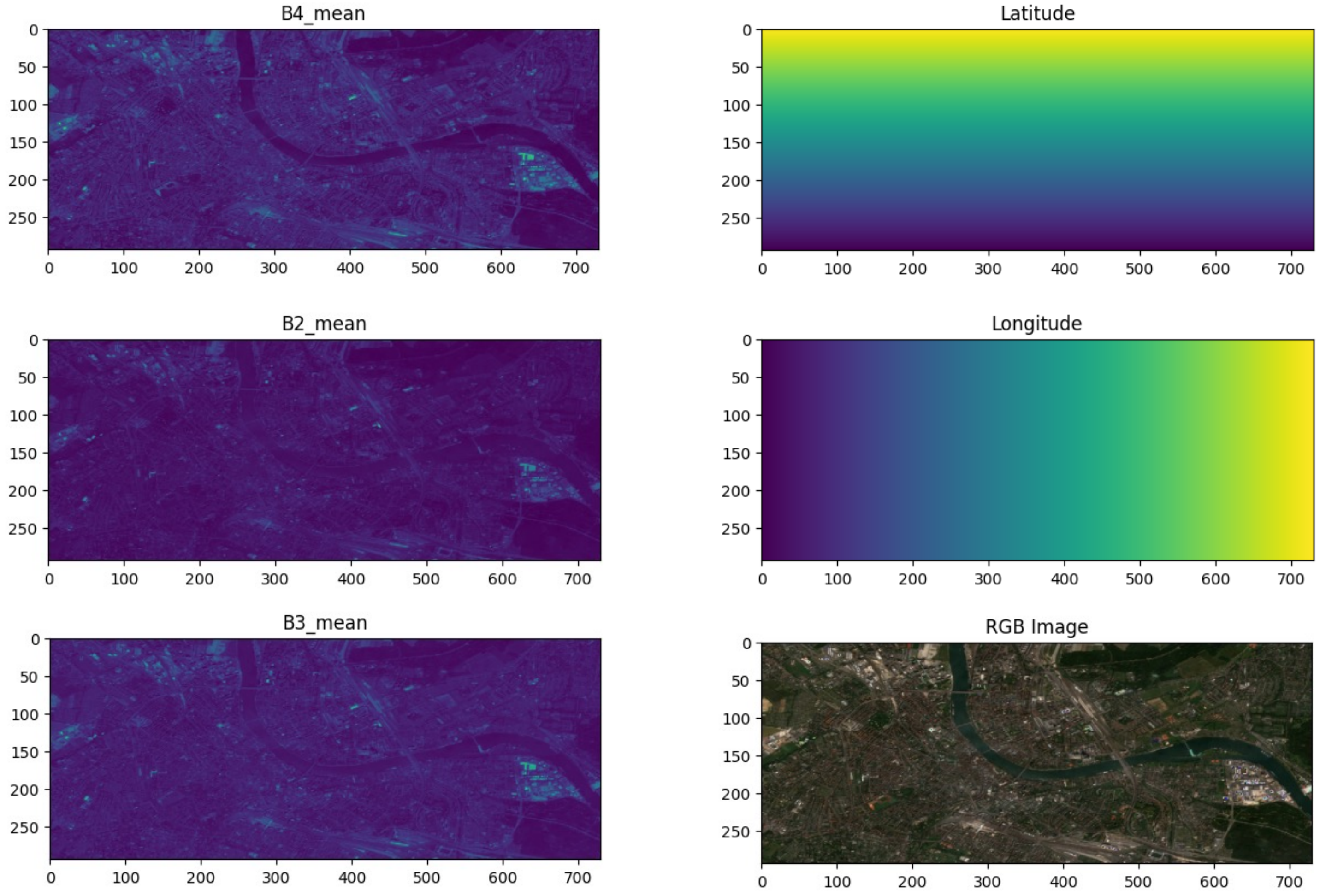
```python
def time_periods_for_date(date: str, num_30_day_periods: int) → ee.Dictionary:
    # ...
    # computes the time periods (e.g.., {"t0": ee.filterDate(t0, t1)})
    # ...

    return ee.Dictionary(results)


def get_time_series_for_point_in_date(
    date: str, latitude: float, longitude: float
) → tuple[list,]:

    point = ee.Geometry.Point([latitude, longitude])
    time_periods = time_periods_for_date(date, 4)

    rain_collection = ee.ImageCollection("ECMWF/ERA5_LAND/HOURLY").select(
        "total_precipitation"
    )

    data = ee.ImageCollection.fromImages(
        time_periods.keys().map(
            lambda time_period: rain_collection.filter(
                ee.Filter(time_periods.get(ee.String(time_period)))
            )
            .reduce(ee.Reducer.sum())
            .rename(ee.String(time_period))
        )
    ).toBands()

    band_arrs = data.sampleRegions(collection=point).getInfo()
    features = band_arrs.get("features", [])

    if len(features) == 0:
        return {}

    return features[-1].get("properties", {})


def enrich_dataframe(dataframe: pandas.DataFrame) → pandas.DataFrame:
    ### ...
    ### Enriches the dataframe with timeseries
    ### ...

    return pandas.concat([dataframe, results], axis=1)


enriched = enrich_dataframe(df)
enriched
```

Compute time periods for each date/point

Fetch the image collection, map through the time periods to filter the data

Merge back to the original dataframe

# Computer Vision | From coordinates to images for Computer Vision applications




- ~275x750 images
- 1 pixel = 15 meters
- Lat and Long images
- RGB reconstructions

```python
# Define Area of Interest
aoi = ee.Geometry.Rectangle(
    [
        [7.549568472036379, 47.5402939467473],
        [7.647758779653566, 47.57979540028916],
    ]
)

# Get the image collection
sentinel_img_collection = (
    ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
    .filterDate("2023-03-01", "2023-08-01")
    .filter(ee.Filter.calendarRange(7, 12, "hour"))
    .filter(ee.Filter.lt("CLOUDY_PIXEL_PERCENTAGE", 20))
    .select(["B4", "B3", "B2"])
)

# Add latitude, longitude, and reproject to scale
sentinel_img = sentinel_img_collection.reduce(ee.Reducer.mean())
sentinel_img = sentinel_img.addBands(sentinel_img.pixelLonLat())
projection = sentinel_img.projection()
sentinel_img = sentinel_img.reproject(crs=projection, scale=15)

# Get the data
bands = list()
sentinel_img_band_names = sentinel_img.bandNames().getInfo()
sentinel_img_band_arrs = sentinel_img.sampleRectangle(
    region=aoi, defaultValue=0
).getInfo()

# Visualize
for band_name in sentinel_img_band_names:
    band = sentinel_img_band_arrs.get("properties").get(band_name)
    np_arr_band = np.array(band)
    plt.imshow(np_arr_band)
    plt.title(band_name.capitalize())
    plt.show()

    if "B" in band_name:
        bands.append(np_arr_band)


rgb_image = np.stack(bands, axis=2)
rgb_img_test = (255 * ((rgb_image - 100) / 3500)).astype("uint8")
plt.imshow(rgb_img_test)
plt.title("RGB Image")
plt.show()
```

Define region of interest (e.g., Basel)

Filter clouds, date and time of day

Define scale, add lat and long

Sample collection in region

Visualize

# You’re not limited to RGB

(you can actually go Multi/Hyperspectral)

# Tricks and tips to keep in mind when working with Geospatial data

It will be painful

Avoid local loops

Learn functional aggregations

Mixing satellites can be tricky

Understand the bands

Dates matter

# Closing thoughts

# Cool. So what?

- More and more data every day
- GEE - a single Python roof
- Tabular, Image, Multispectral
- Community is great (GeeMap, MapScaping podcast)
- Sustainability use cases
- Our planet tells a story

# Thanks! 🇨🇭