Intro

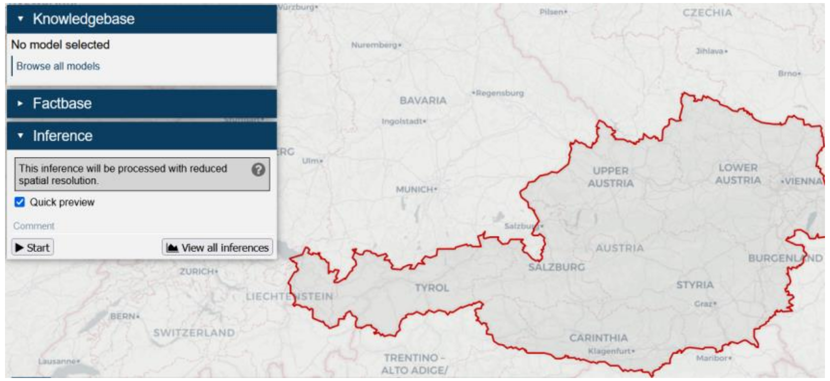
The following set of slides gives an introduction to the functionality and usage of gsemantique. It starts with placing gsemantique in the context of foundational works, i.e. mainly the semantique package.

Foundations and extensions of sen2cube

sen2cube.at

The Austrian semantic data cube

- comprehensive, technology-stack spanning application (SaaS)
- preconfigured data sets with country-level coverage
- graphical user interface



SEMANTIQUE

framework for semantic querying in EO data cubes

gsemantique

Ad-hoc data cubes anywhere on Earth

- standalone python package
- data sets with global coverage
- internal chunking of spatio-temporal extents
- executable locally or in the cloud
- customisable and extensible

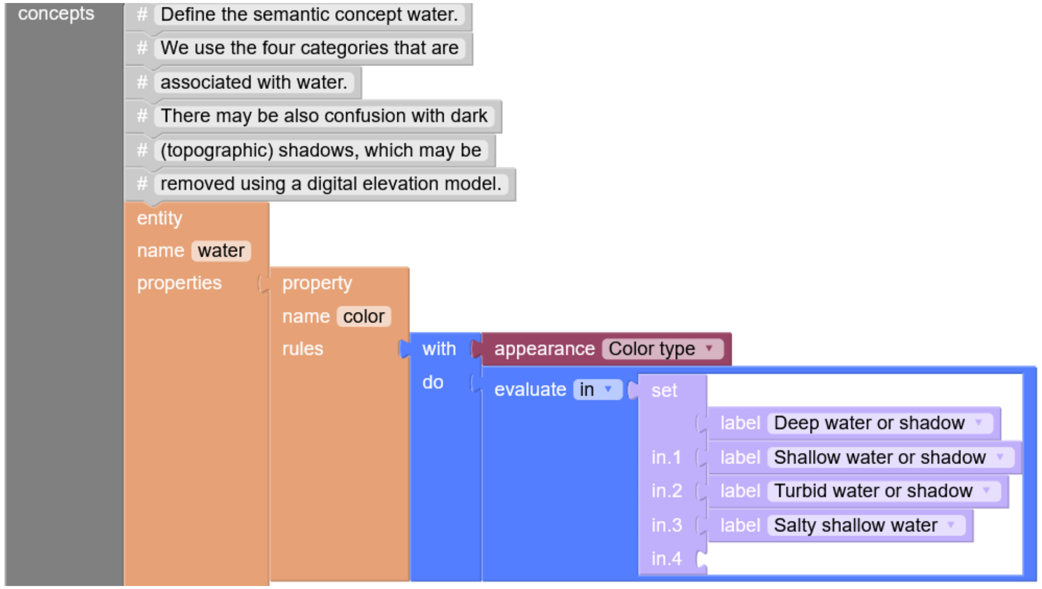
```
mapping = sq.mapping.Semantique()
mapping["entity"] = {}
mapping["entity"]["vegetation"] = {
    "class": sq.layer("Planet", "classification", "scl").evaluate("equal", 4)
}
mapping["entity"]["cloud_snow"] = {
    "class": sq.layer("Planet", "classification", "scl").evaluate("in", [8,9,10,11])
}

context = {
    "datacube": dc,
    "mapping": mapping,
    "space": space,
    "time": time,
    "crs": epsg,
    "spatial_resolution": [-res, res]
}
```

Semantique

SEMANTIQUE

- framework for semantic querying
- underlying processing engine in sen2cube.at
- implemented as an open-source Python package

Sen2cube view



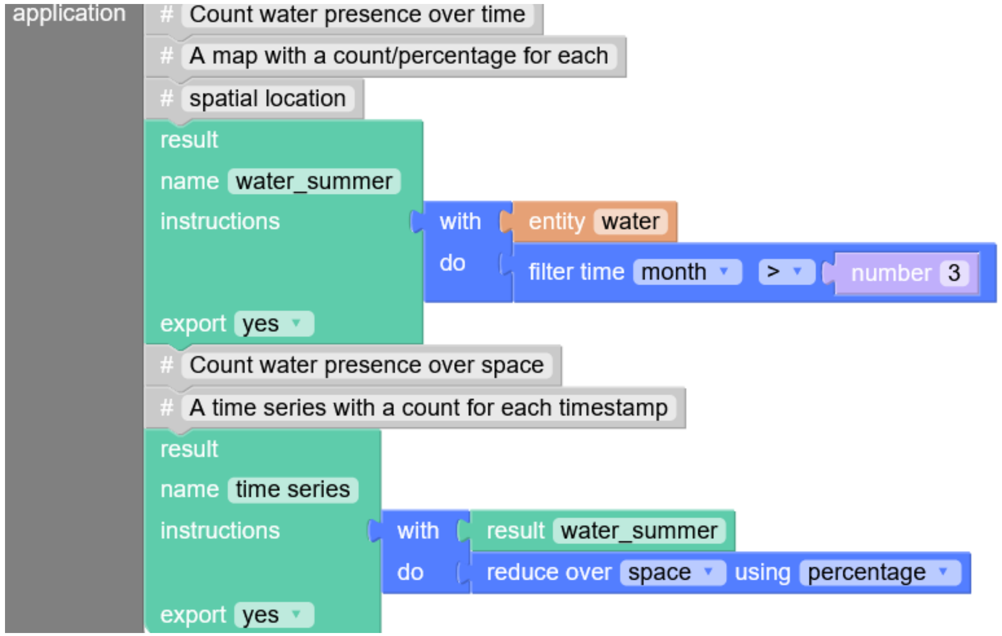
Semantique view

```python
mapping = sq.mapping.Semantique()
mapping["entity"] = {}
mapping["entity"]["water"] = {
    "color": (
        sq.appearance("colortype")
          .evaluate("in", [21, 22, 23, 24])
    )
}
```

Semantique

- framework for semantic querying
- underlying processing engine in sen2cube.at
- implemented as an open-source Python package

Sen2cube view

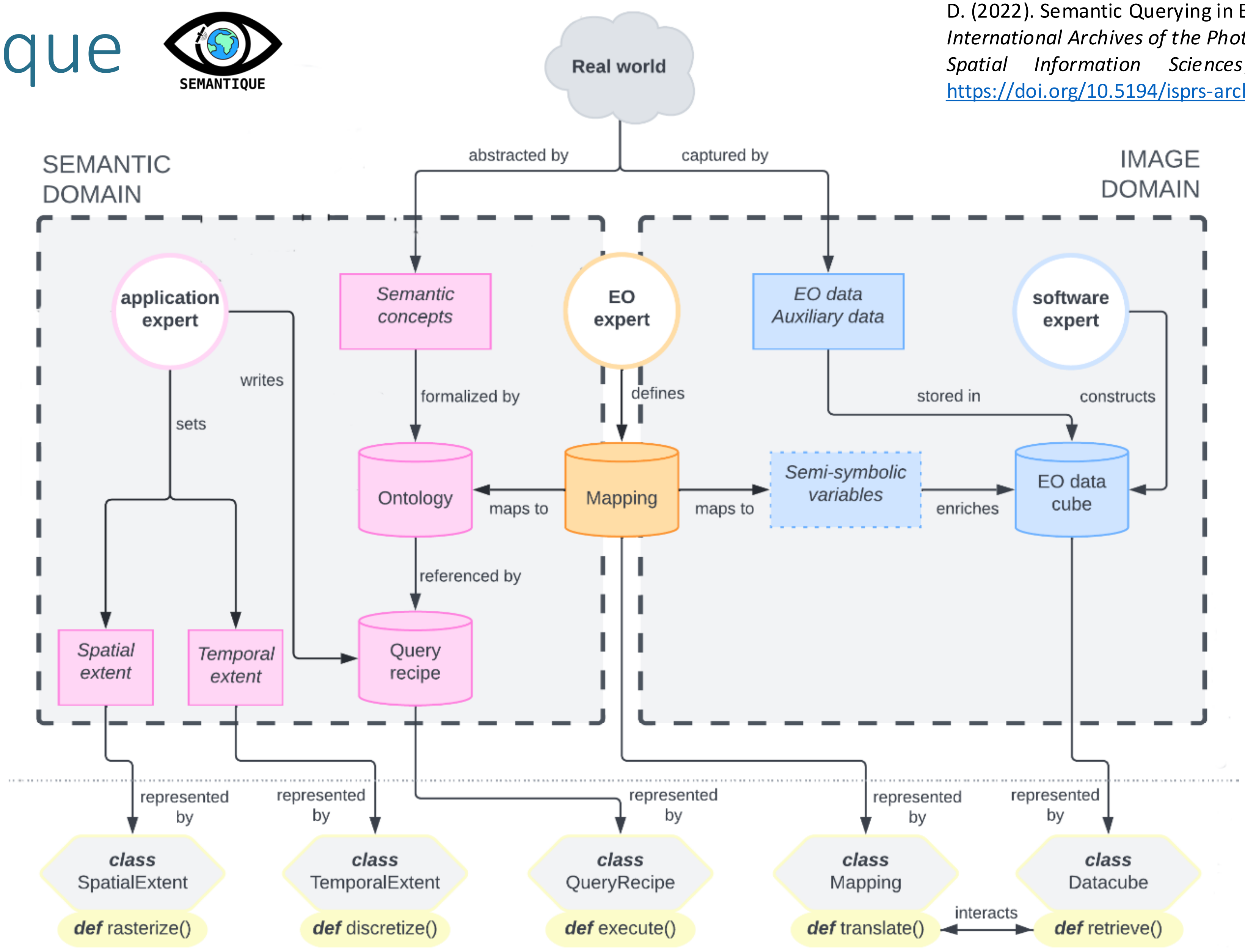


Semantique view

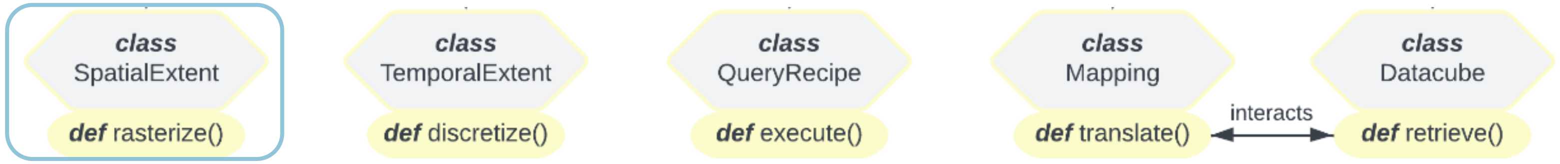
```
recipe = sq.QueryRecipe()
recipe["water_summer"] = (
    sq.entity("water")
    .filter_time("month", "greater", 3)
)
recipe["time series"] = (
    sq.result("water_summer")
    .reduce("percentage", "space")
)
```

Semantique

Van Der Meer, L., Sudmanns, M., Augustin, H., Baraldi, A., & Tiede, D. (2022). Semantic Querying in Earth Obsevration Data Cubes. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLVIII-4/W1-2022*, 503–510. https://doi.org/10.5194/isprs-archives-XLVIII-4-W1-2022-503-2022



Semantique – individual components



- spatial extents can be bboxes, single or multiple features
- based on geopandas package (GeoDataFrame initializer)

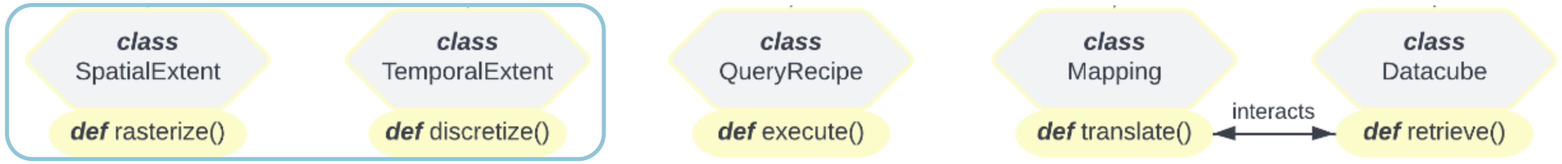
```python
import geopandas as gpd
import semantique as sq
from shapely.geometry import box

# define spatial extent from geojson file
gdf = gpd.read_file("files/footprint.geojson")
space = sq.SpatialExtent(gdf)

# define spatial extent from bbox coords
xmin, ymin, xmax, ymax = 9,50,10,51
aoi = box(xmin, ymin, xmax, ymax)
gdf = gpd.GeoDataFrame(geometry=[aoi], crs=4326)
space = sq.SpatialExtent(gdf)
```

Semantique – individual components

- temporal extents can be datetime strings (start & end)
- based on pandas package (Timestamp initializer)

```
import semantique as sq
time = sq.TemporalExtent("2019-01-01", "2020-12-31")
```

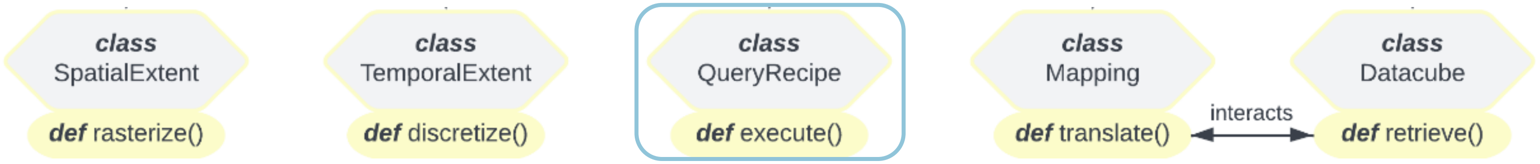
Semantique – individual components



- full description of spatio-temporal extents requires some additional information to be specified
  - spatial resolution *(required)*
  - coordinate reference system *(optional)*
  - timezone *(optional)*

```python
from semantique.processor.utils import parse_extent

extent = parse_extent(
    space,
    time,
    spatial_resolution = [-10, 10],
    crs = 3035,
    tz = "UTC"
)
```

Semantique – individual components



- application model references semantic concepts
- based on a Python dictionary

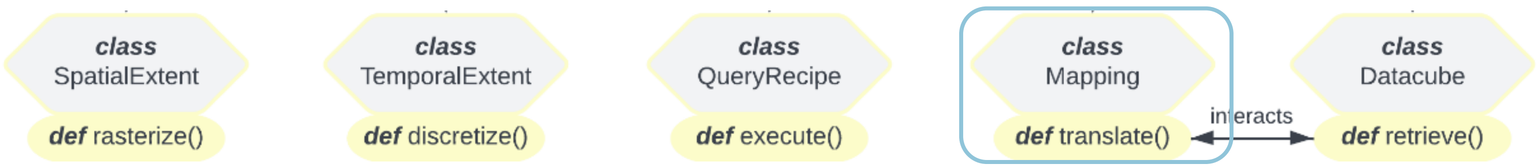
```python
import semantique as sq

recipe = sq.QueryRecipe()
recipe["water_summer"] = (
    sq.entity("water")
    .filter_time("month", "greater", 3)
)
recipe["time series"] = (
    sq.result("water_summer")
    .reduce("percentage", "space")
)
```

```python
import json

# to persist the recipe on disk
with open("recipe.json", "w") as file:
    json.dump(recipe, file, indent = 2)

# to read it back into memory
with open("recipe.json", "r") as file:
    recipe = sq.QueryRecipe(json.load(file))
```

recipe.json

```json
{
  "water_summer": {
    "type": "processing_chain",
    "with": {
      "type": "concept",
      "reference": [
        "entity",
        "water"
      ]
    },
    "do": [
      {
        "type": "verb",
        "name": "filter",
        "params": {
          "filterer": {
            "type": "processing_chain",
            "with": {
              "type": "self"
            },
            ...
```

Semantique – individual components



- mapping links each semantic concept to data values in an EO data cube
- based on a Python dictionary

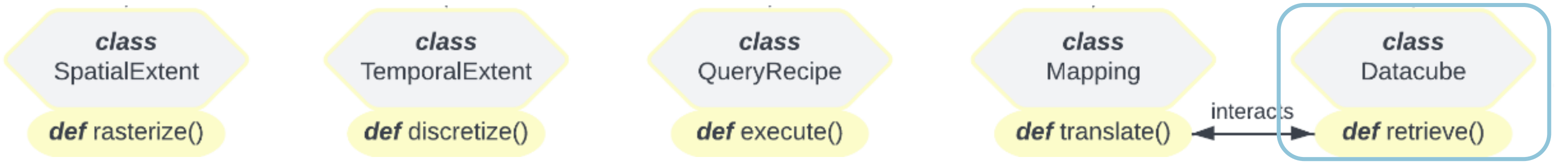
```python
import semantique as sq

mapping = sq.mapping.Mapping()
mapping["entity"] = {}
mapping["entity"]["water"] = {
    "color": (
        sq.appearance("colortype")
          .evaluate("in", [21, 22, 23, 24])
    )
}
```

```python
import json

# to persist the mapping on disk
with open("mapping.json", "w") as file:
    json.dump(mapping, file, indent = 2)

# to read it back into memory
with open("mapping.json", "r") as file:
    mapping = sq.mapping.Semantique(json.load(file))
```

mapping.json

```json
{
  "entity": {
    "water": {
      "color": {
        "type": "processing_chain",
        "with": {
          "type": "layer",
          "reference": [
            "appearance",
            "colortype"
          ]
        },
        "do": [
          {
            "type": "verb",
            "name": "evaluate",
            "params": {
              "operator": "in",
              "y": {
                "type": "set",
                "content": [
                  21,
                  ...
```

Semantique – individual components



- multi-dimensional array structure storing the data
- 3 different configurations of data cube objects available

How is the cube is organized?

(a)

enables to interact with zipped archive of multiple GeoTIFF files

only for demo purposes

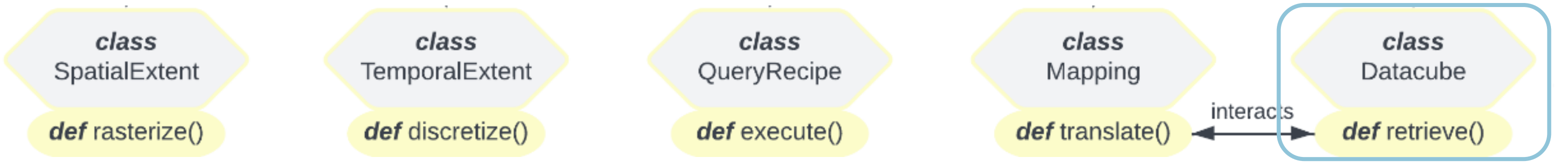
(b)

enables to interact with EO data cube instances deployed using ODC

(c)

enables to interact with STAC metadata search results

Semantique – individual components



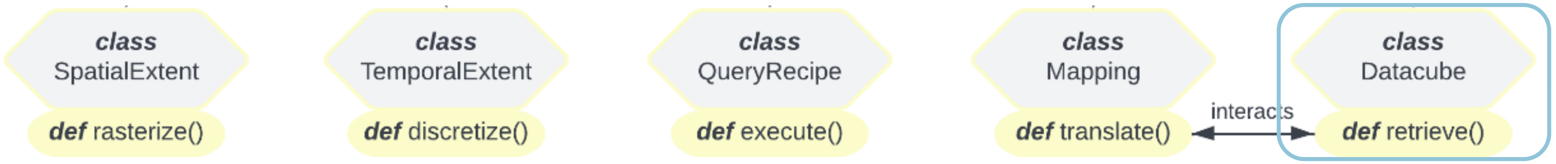
- all data cubes are represented by their layout
- layout intended to be created once by software/EO expert
- specific layout attributes exist for individual data cube configurations
- based on a Python dictionary & stored as a JSON-structured file

```json
{
    "reflectance": {
        "s2_band01": {
            "name": "coastal",
            "description": "Coastal aerosol band (~443nm; 60m spatial
            resolution). Top of atmosphere (TOA) reflectance captured by
            the multi-spectral instrument (MSI) of Sentinel-2A or
            Sentinel-2B as band 1.",
            "type": "continuous",
            "values": {
                "min": 0,
                "max": 1,
                "precision": 1
            },
            "file": "coastal.geotiff",
            "copyright": "Contains modified Copernicus data."
        },
        "s2_band02": {
            "name": "...",
```

layout.json
(GeoTiffArchive)

```python
import json
import semantique as sq

with open("files/layout_gtiff.json", "r") as file:
    dc = sq.datacube.GeotiffArchive(
        json.load(file),
        src = "files/layers_gtiff.zip"
    )
```

Semantique – individual components



- all data cubes are represented by their layout
- layout intended to be created once by software/EO expert
- specific layout attributes exist for individual data cube configurations
- based on a Python dictionary & stored as a JSON-structured file

```json
{
    "reflectance": {
        "s2_band01": {
            "name": "coastal",
            "description": "Coastal aerosol band (~443nm; 60m spatial resolution). Top of atmosphere (TOA) reflectance captured by the multi-spectral instrument (MSI) of Sentinel-2A or Sentinel-2B as band 1.",
            "type": "continuous",
            "values": {
                "min": 0,
                "max": 1,
                "precision": 1
            },
            "dtype": "float32",
            "na_value": "NA",
            "copyright": "Contains modified Copernicus data."
        },
        "s2_band02": {
            "name": "..."
```

layout.json
(STACCube)

```python
import pystac
import semantique as sq
from pystac_client import Client
from shapely.geometry import box

# define temporal & spatial range to perform STAC query
xmin, ymin, xmax, ymax = -2.75, 47.25, -2.25, 47.75
aoi = box(xmin, ymin, xmax, ymax)
t_range = ["2020-07-15", "2020-08-01"]

# STAC-based metadata retrieval
catalog = Client.open("https://earth-search.aws.element84.com/v1")
query = catalog.search(
    collections="sentinel-2-l2a",
    datetime=t_range,
    limit=100,
    intersects=aoi
)
item_coll = query.item_collection()

# define datacube
with open("files/layout_stac.json", "r") as file:
    dc = sq.datacube.STACCube(
        json.load(file),
        src = item_coll
    )
```

Semantique – Hands-on

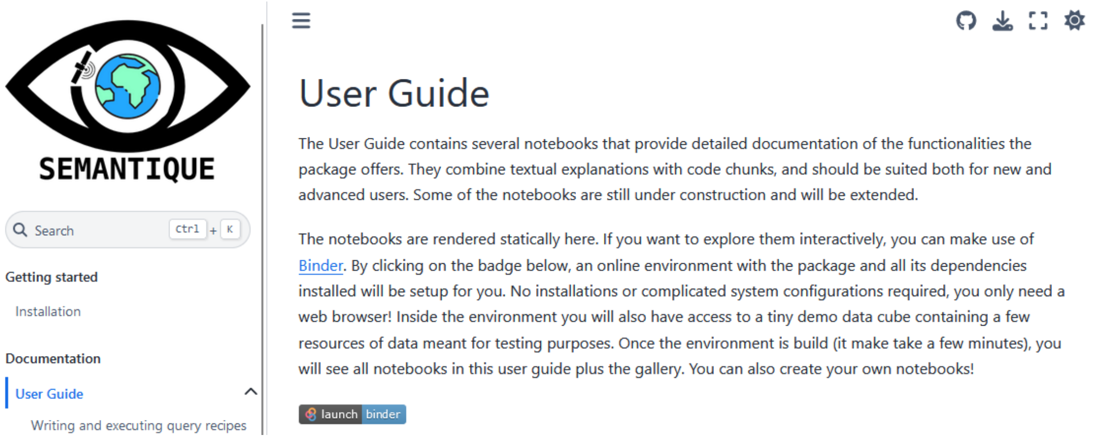
README

Semantique tutorials

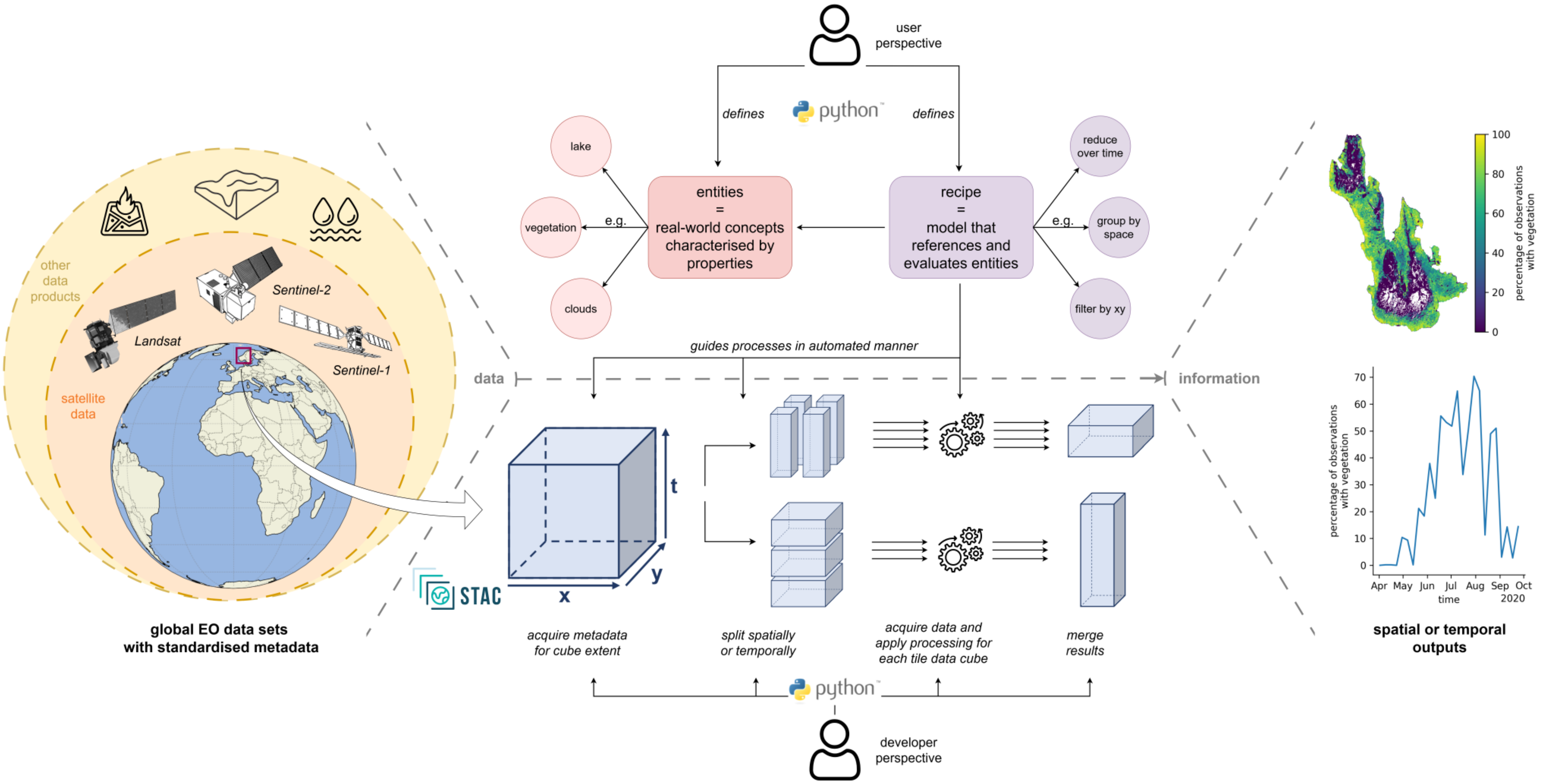
notebook	link	description
`recipes.ipynb`	Open in Colab	explaining semantique's basics
`gallery.ipynb`	Open in Colab	showcasing different application examples

Two example notebooks
https://github.com/fkroeber/semantique_tutorials

Additional help/information
https://github.com/ZGIS/semantique



Gsemantique: On-demand EO data cubes



Gsemantique: On-demand EO data cubes

Core functionalities (I)

✓ Pre-configured access to a variety of EO datasets with global coverage (e.g. Sentinel-1, Sentinel-2, Landsat)

provider	collection	category	temporality
Planet	sentinel-1-rtc	SAR	s
Planet	sentinel-2-l2a	multispectral	s
Planet	landsat-c2-l2	multispectral	s
Planet	esa-worldcover	landcover	Y
Planet	io-lulc-annual-v02	landcover	Y
Planet	nasadem	DEM	None
Planet	cop-dem-glo-30	DSM	None
Planet	modis-64A1-061	fire detection	M
Planet	modis-14A2-061	fire detection	D
Planet	jrc-gsw	hydrogeography	None
Element84	sentinel-2-l2a	multispectral	s
ASF	sentinel-1-global-coherence	SAR	3M
ASF	glo-30-hand	hydrogeography	None

Gsemantique: On-demand EO data cubes

Core functionalities (I)

✓ Pre-configured access to a variety of EO datasets with global coverage (e.g. Sentinel-1, Sentinel-2, Landsat)

semantique

```
import pystac
import semantique as sq
from pystac_client import Client
from shapely.geometry import box

# define temporal & spatial range to perform STAC query
xmin, ymin, xmax, ymax = -2.75, 47.25, -2.25, 47.75
aoi = box(xmin, ymin, xmax, ymax)
t_range = ["2020-07-15", "2020-08-01"]

# STAC-based metadata retrieval
catalog = Client.open("https://earth-search.aws.element84.com/v1")
query = catalog.search(
    collections="sentinel-2-l2a",
    datetime=t_range,
    limit=100,
    intersects=aoi
)
item_coll = query.item_collection()

# define datacube
with open("files/layout_stac.json", "r") as file:
    dc = sq.datacube.STACCube(
        json.load(file),
        src = item_coll
    )
```

gsemantique

```
import json
import os
import semantique as sq
import gsemantique as gsq
from gsemantique.data.search import Finder
```

Layout file

```
# get layout file
package_dir = os.path.split(gsq.__file__)[0]
layout_path = os.path.join(package_dir, "data", "layout.json")
```

```
# search for items in recipe
with open(layout_path, "r") as file:
    dc = sq.datacube.STACCube(
        json.load(file),
        src = [],
    )
```

Finder class

```
fdr = Finder(ds_catalog, t_start, t_end, aoi_bbox)
fdr.search_auto(recipe, mapping, dc)
```

```
# define datacube
with open(layout_path, "r") as file:
    dc = sq.datacube.STACCube(
        json.load(file),
        src = fdr.item_coll
    )
```

Gsemantique: On-demand EO data cubes

Core functionalities (II)

✓ Retrieval & storage mechanisms for the data to persist data cube inputs locally

Downloader class

```
# download items that have been found
out_dir = "files/data"
dwn = Downloader(fdr.item_coll, out_dir)
dwn.run()
```

```
landsat-c2-l2 (collection 1/1)
Not enough items to estimate size. Skipping preview run.
Downloading EO data: 168MB [00:18, 9.55MB/s]
```

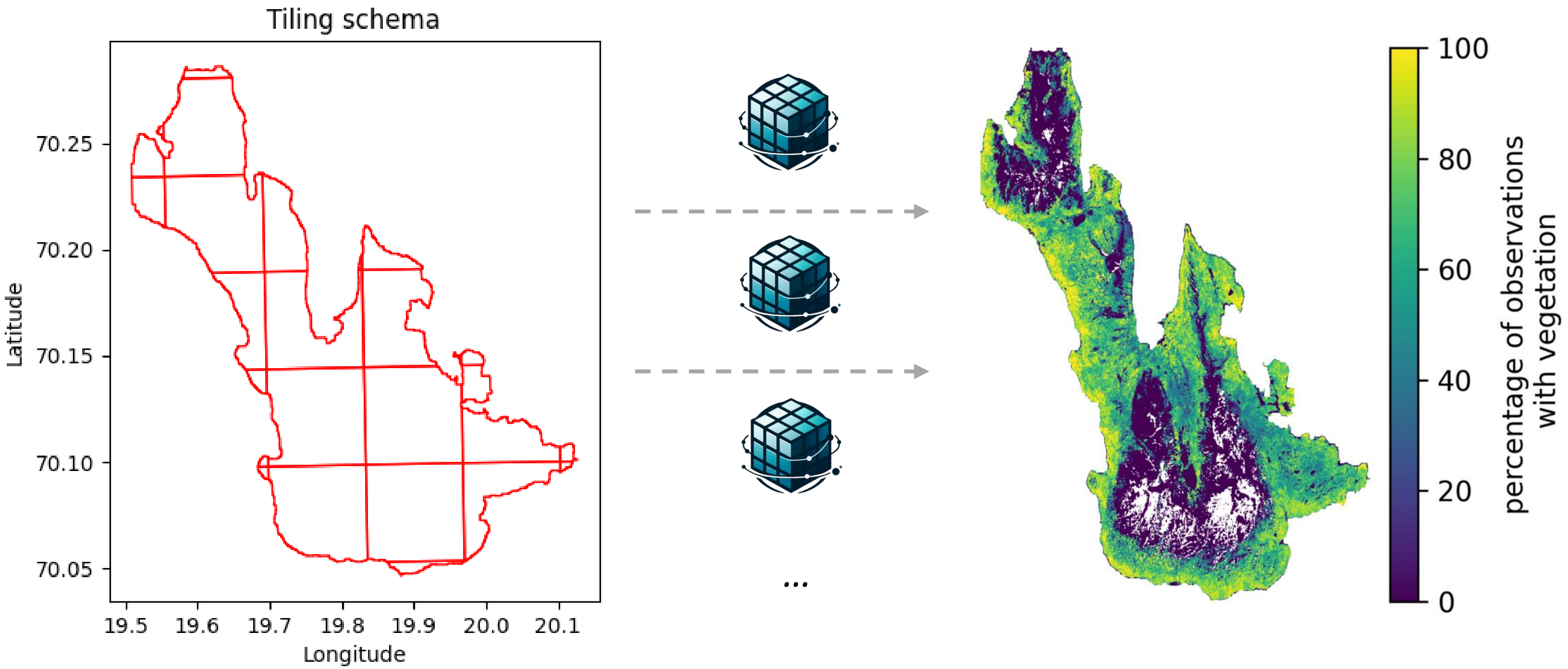
Gsemantique: On-demand EO data cubes

Core functionalities (III)

Scaling mechanisms that allow to evaluate recipes for large spatio-temporal extents up to the mesoscale

- automated analysis of recipe for possible tiling dimension
- sequential execution of recipe & merging of results aftwerwards

e.g. for spatial tiling



Gsemantique: On-demand EO data cubes

Core functionalities (III)

✓ Scaling mechanisms that allow to evaluate recipes for large spatio-temporal extents up to the mesoscale

semantique

```python
context = {
    "datacube": dc,
    "mapping": mapping,
    "space": space,
    "time": time,
    "crs": 3035,
    "tz": "UTC",
    "spatial_resolution": [-10, 10]
}

response = recipe.execute(**context)
```

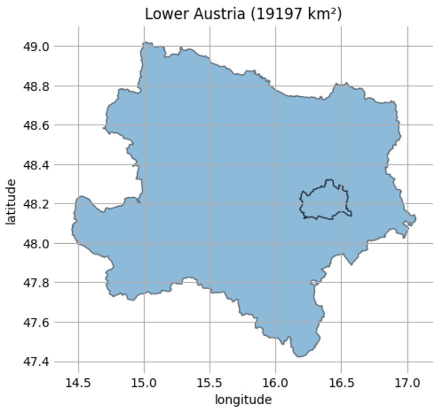
gsemantique

```python
# create TileHandler & execute recipe
context = {
    "recipe": recipe,
    "datacube": dc,
    "mapping": mapping,
    "space": space,
    "time": time,
    "crs": 3035,
    "tz": "UTC",
    "spatial_resolution": [-10, 10],
    "chunksize_s": 1024,
    "tile_dim": None,
    "merge_mode": 'merged',
    "out_dir": None,
    "reauth": True,
}

th = TileHandler(**context)
th.execute()
```

optional arguments

TileHandler class

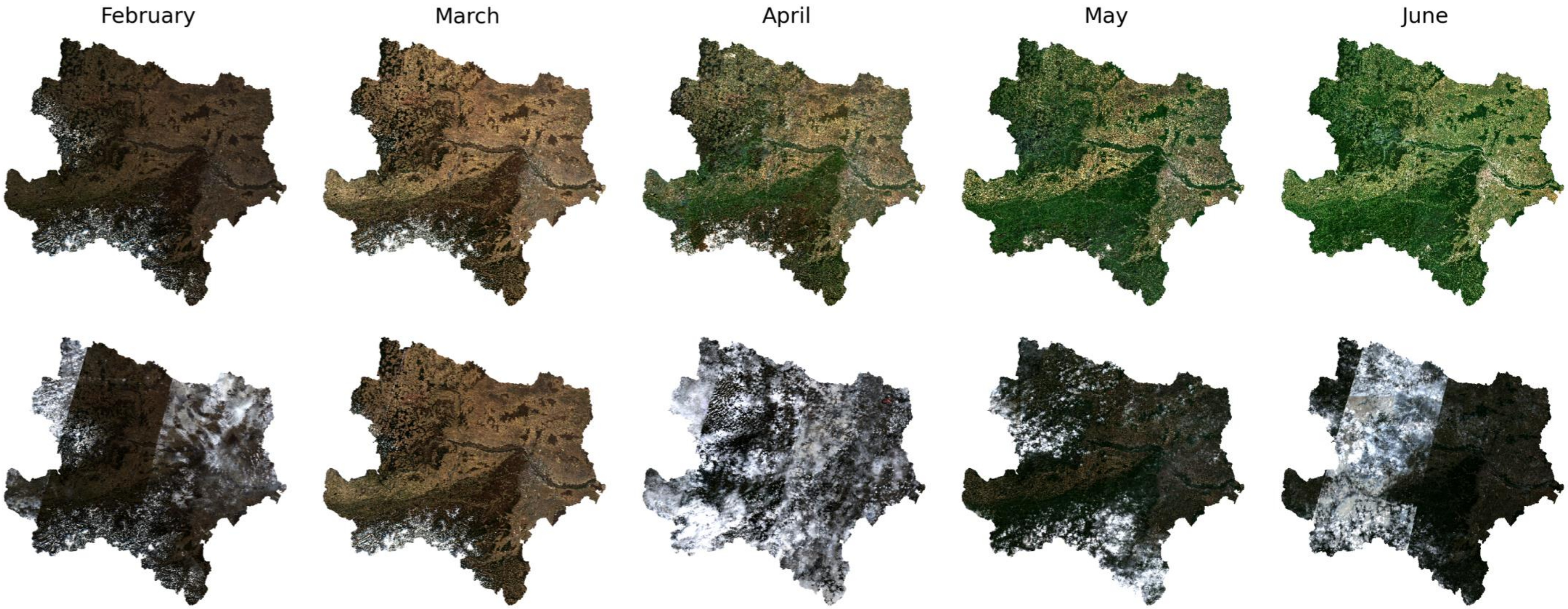
What can semantic, on-demand EO cubes be used for?



Cloud-free composites

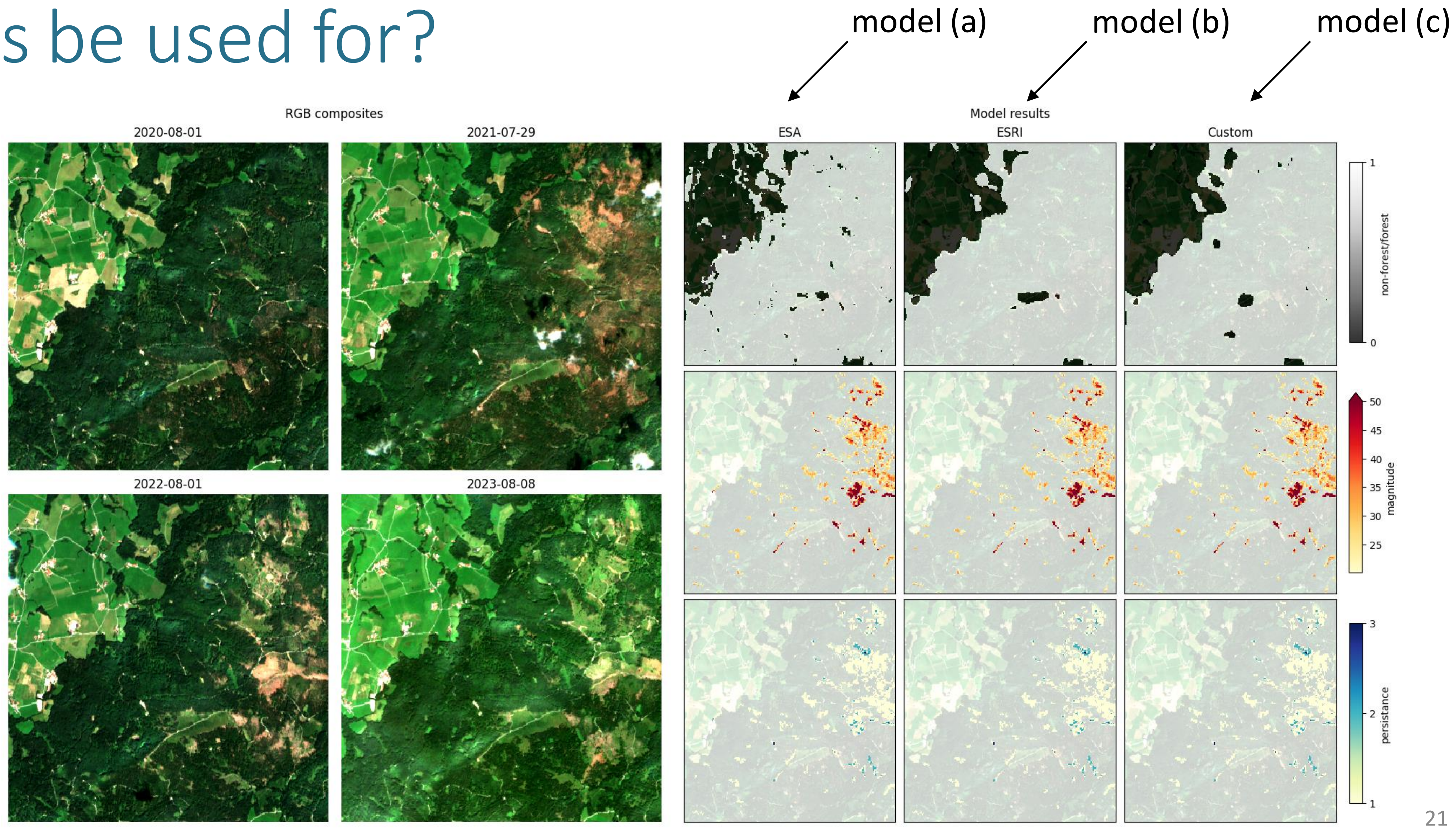
semantic querying

median-composite



What can semantic, on-demand EO cubes be used for?

Forest disturbance detection
–
Model comparisons



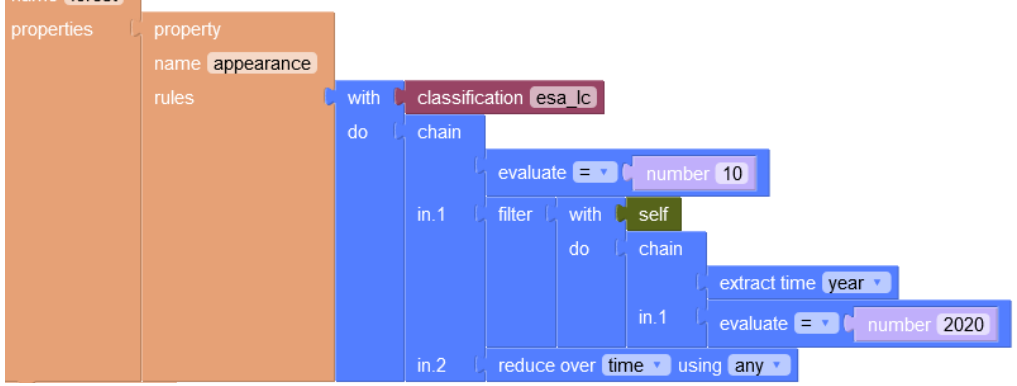
What can semantic, on-demand EO cubes be used for?

Forest disturbance detection

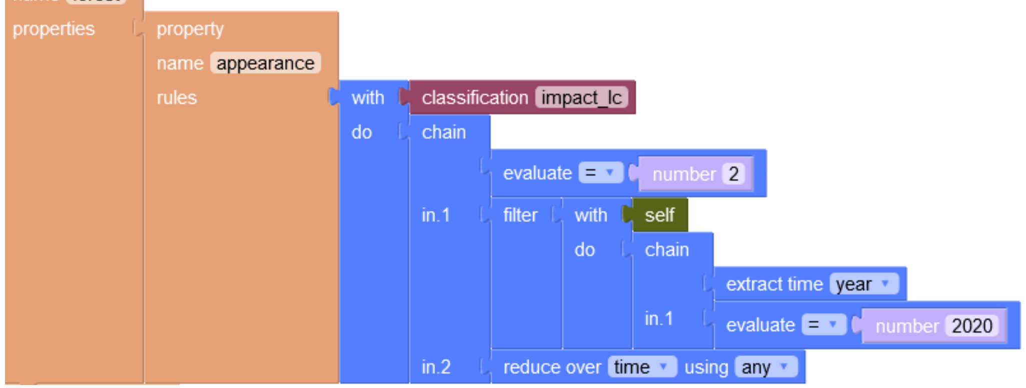
–

Model comparisons

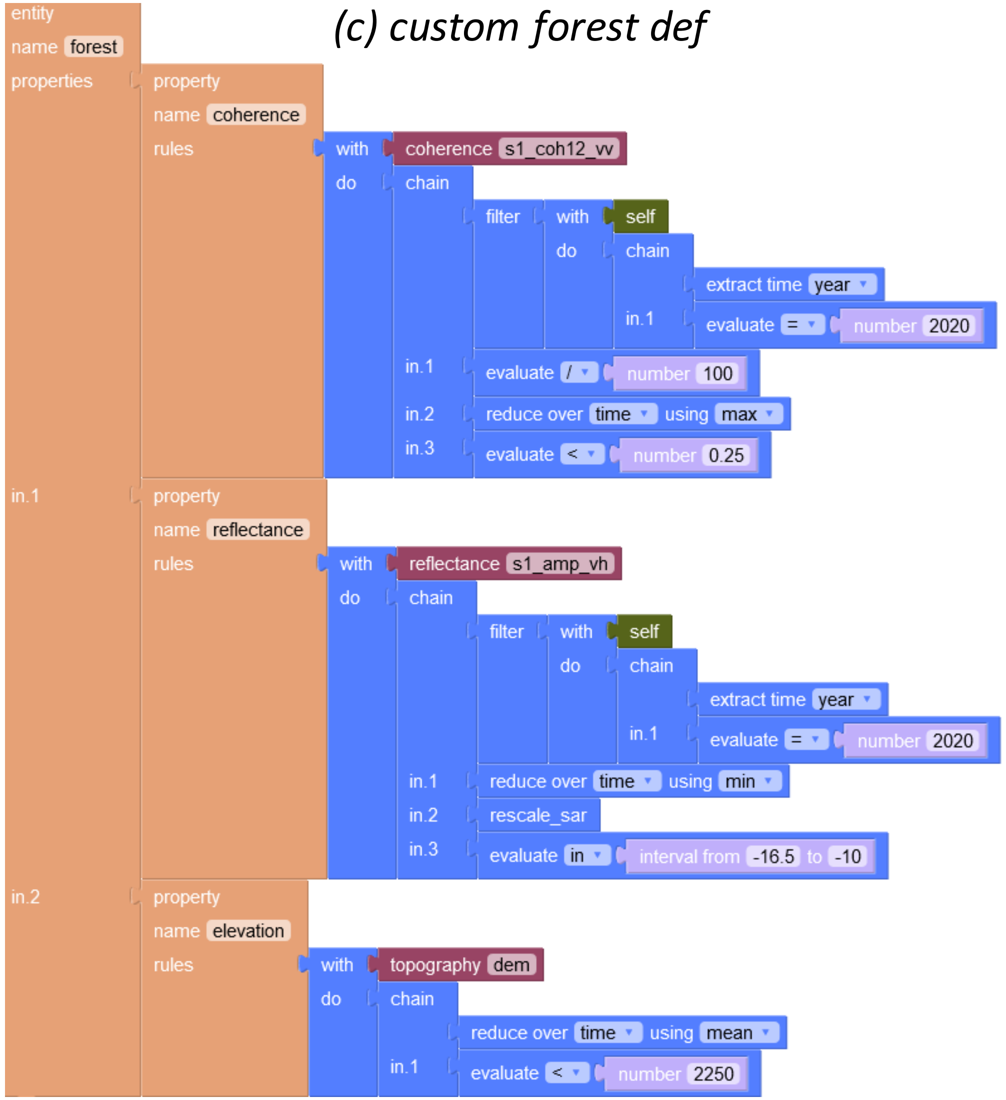
(a) ESA forest def



(b) ESRI forest def



(c) custom forest def

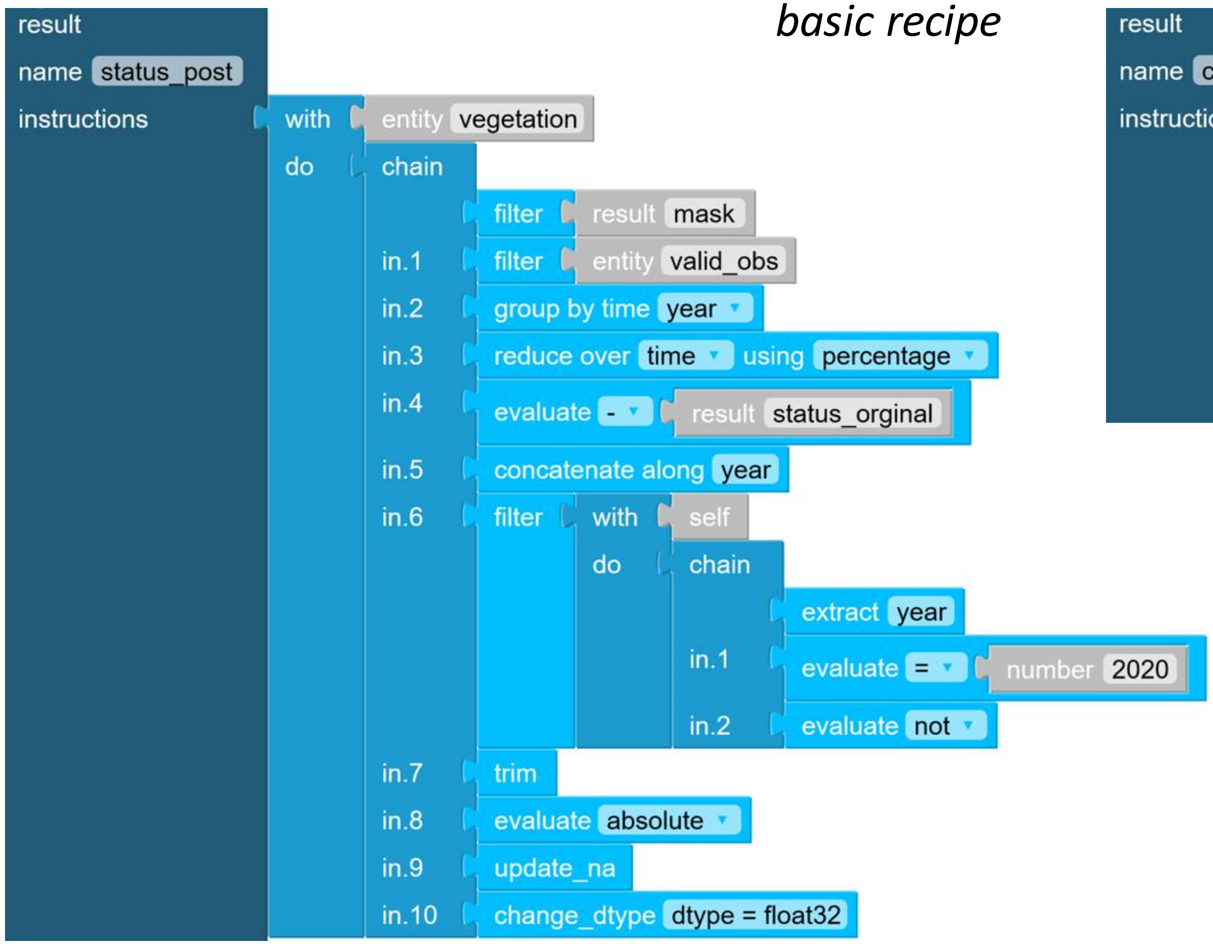


What can semantic, on-demand EO cubes be used for?

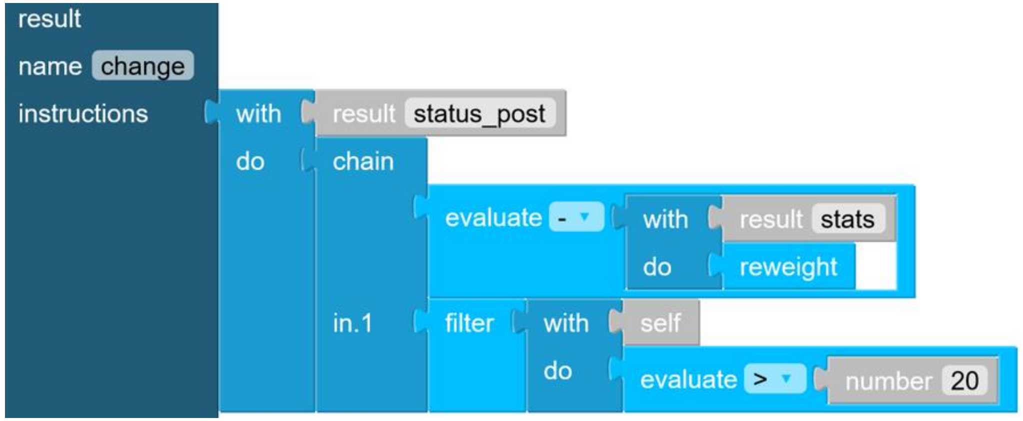
Forest disturbance detection

–

Model comparisons

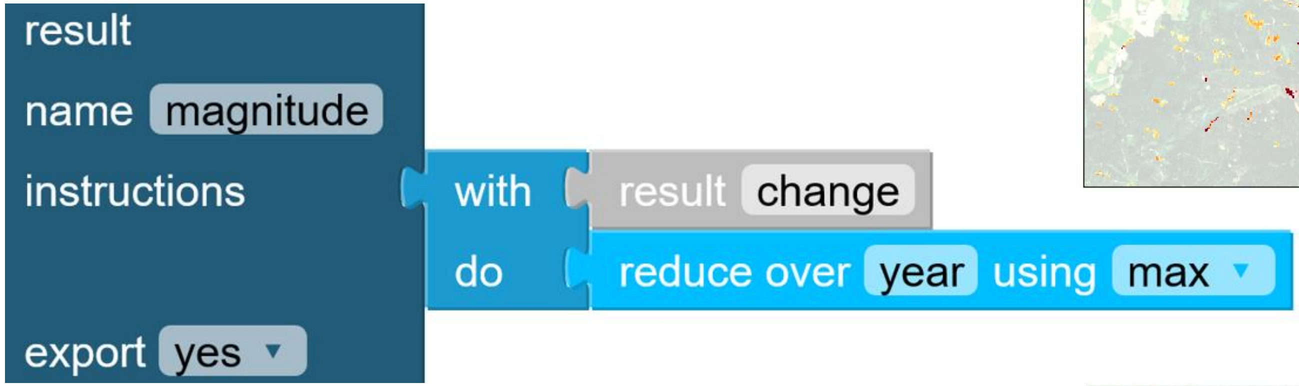


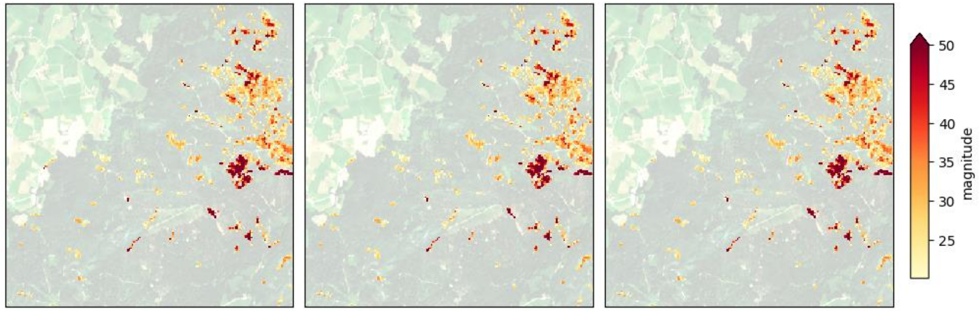
basic recipe

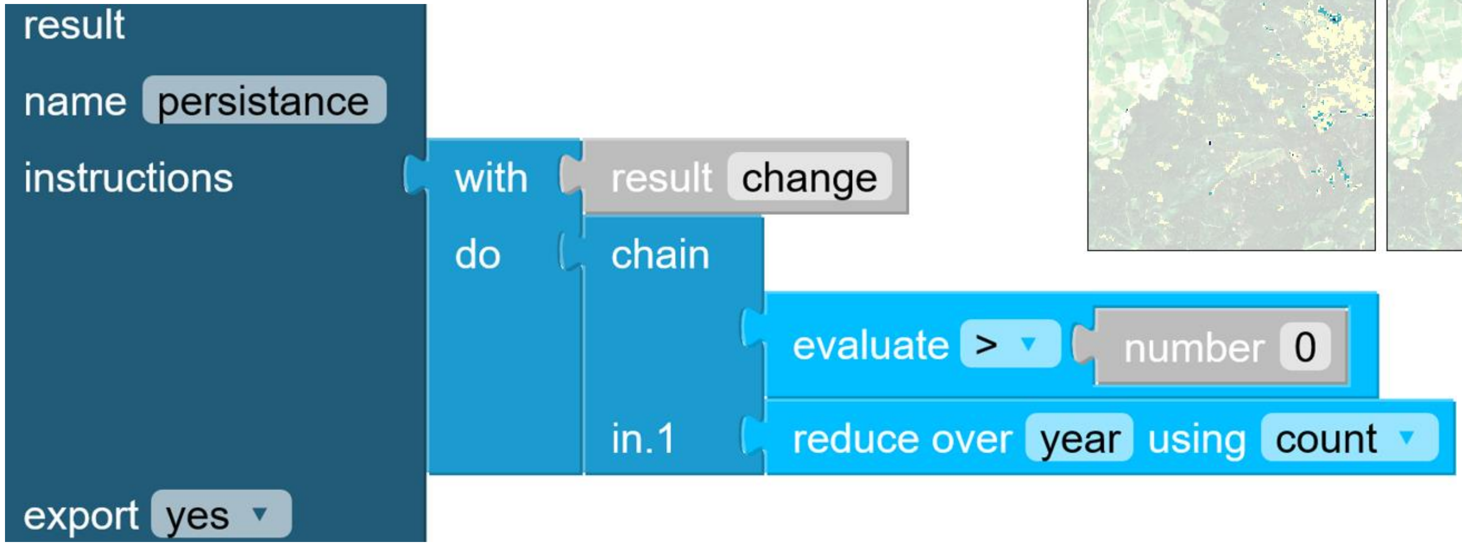


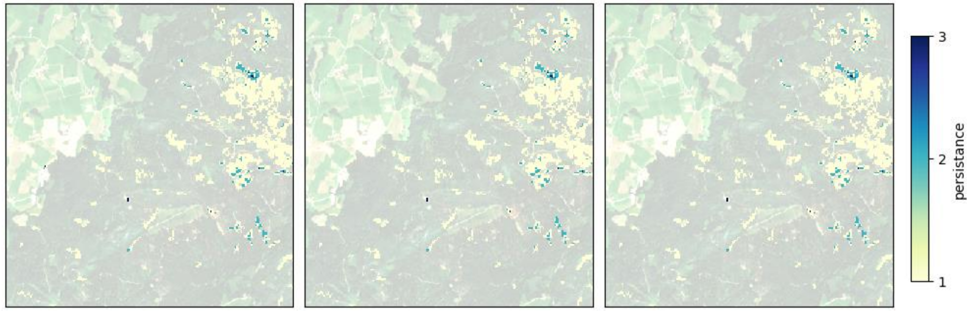
What can semantic, on-demand EO cubes be used for?

Forest disturbance detection
–
Model comparisons

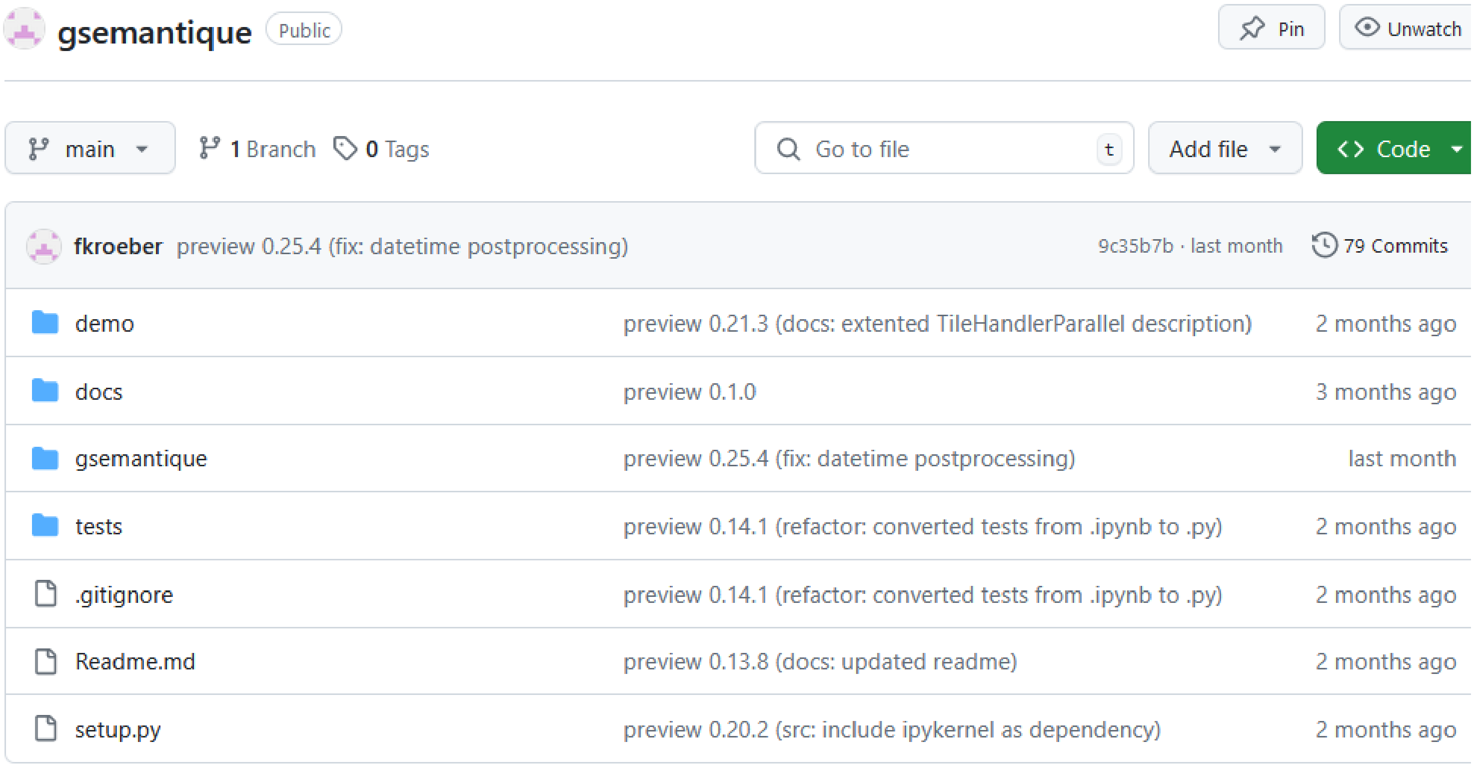








Gsemantique – Hands-on



Three example notebooks
https://github.com/fkroeber/gsemantique