Introdution to Vector Data Cubes

In this notebook, we convert a raster data cube covering a large spatial extent to a vector data cube containing only the spatial areas of interest (AOI). We also observe how the geospatial features, and data associated with those features, change over time and compute their zonal statistics. In this case study, our study area is the Annapurna Conservation Area in Nepal. Our vector data is the forest class from the land use and land cover of the conservation area as a shapefile and we obtain MODIS FPAR for the AOI, our raster data, using the Spatio Temporal Asset Catalog (STAC).

Here we present the code for analysis in R and Python.

What is a Vector Data Cube?

Vector data cubes are n-D arrays with (at least) one dimension that maps to a set of typically 2-D vector geometries (points, lines or polygons). Examples include - Time series of weather measurements for a set of stations, - Population indicators for countries in Africa, or - Statistics of indicators for different land use classes in Germany.

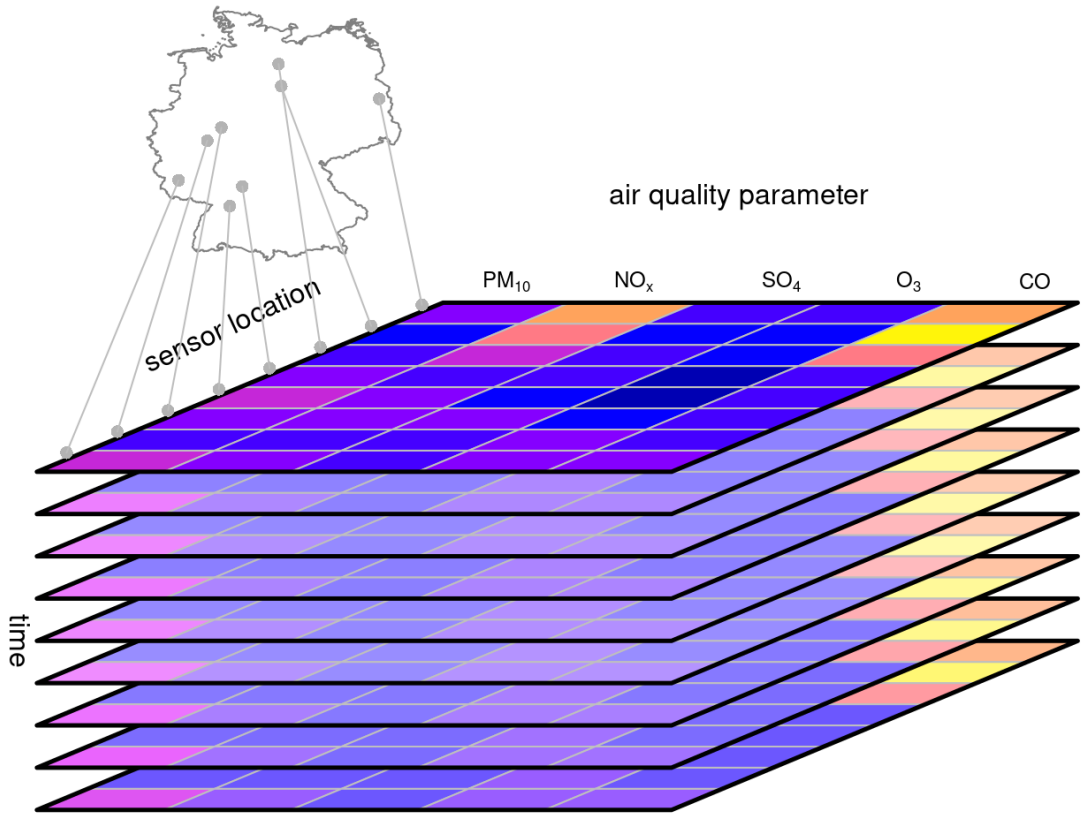


Figure 1: Visual representation of a vector data cube.

Loading the necessary packages

R

```
library(sf)
library(dplyr)
library(ggplot2)
```

Python

```
import pystac_client
import planetary_computer
import folium
import geopandas as gpd
import leafmap.foliumap as leafmap
```

Open the shapefile for the AOI

We want to open the shapefile of the land use area for the conservation area. We are using a subset of the shapefile of the forest area to construct a vector datacube. We take the bbox to use as a search input in the selection of the earth observation data.

R

```
# Read the shapefile

nepal <- st_read("Landuse-Annapurna_conservation_Area_NEPAL/Landuse.shp")
```

```
Reading layer `Landuse' from data source
  `/home/fcremer/Documents/NFDI4Earth/DataCubes_Tutorial/Landuse-Annapurna_conservation_Area
  using driver `ESRI Shapefile'
Simple feature collection with 4035 features and 7 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 83.47956 ymin: 28.24066 xmax: 84.46899 ymax: 29.33133
Geodetic CRS:  WGS 84
```

```
# Nepal bbox
nepal_bbox <- nepal |>
  sf::st_transform(4326) |>
  sf::st_bbox()
```

```r
# Filter the forest
forest <- nepal %>%
  sf::st_transform(st_crs(nepal_bbox)) |>
  filter(Category == "Forest")
```

Python

```python
nepal = gpd.read_file("Landuse-Annapurna_conservation_Area_NEPAL/Landuse.shp")

bbox = nepal.total_bounds

forest = nepal[nepal["Category"]=="Forest"].reset_index(drop=True)
forest.head(3)
```

```
    fid    FCODE  ... LakeName                                          geometry
0  2666  25212.0  ...     None  POLYGON ((83.86829 28.3053, 83.86844 28.30611,...
1   789  25212.0  ...     None  POLYGON ((83.67782 28.83034, 83.67792 28.83043...
2   792  25212.0  ...     None  POLYGON ((83.68051 28.83133, 83.68051 28.83129...

[3 rows x 8 columns]
```

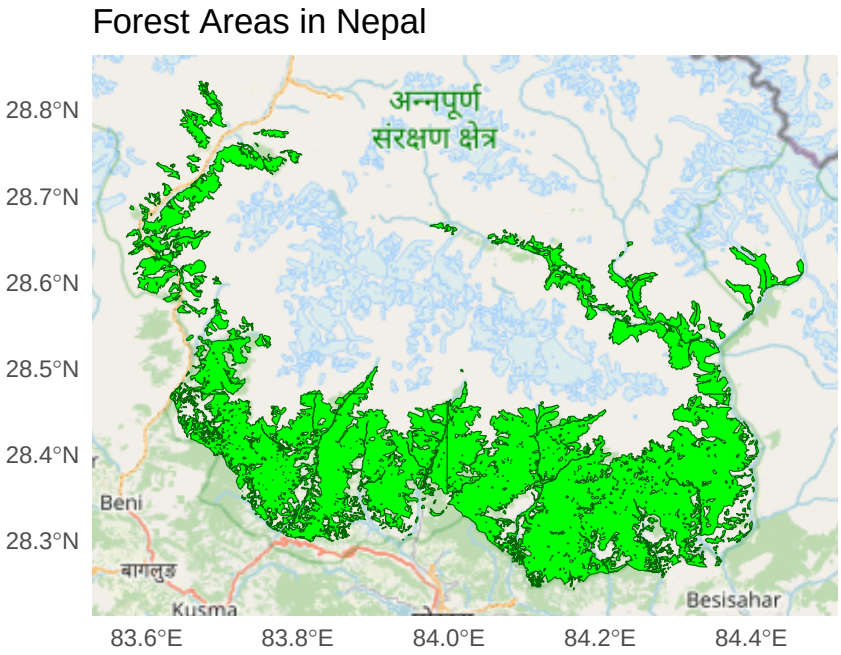
Plot of the Forest areas

Now that we loaded the data, we can have a look at the forest polygons.

R

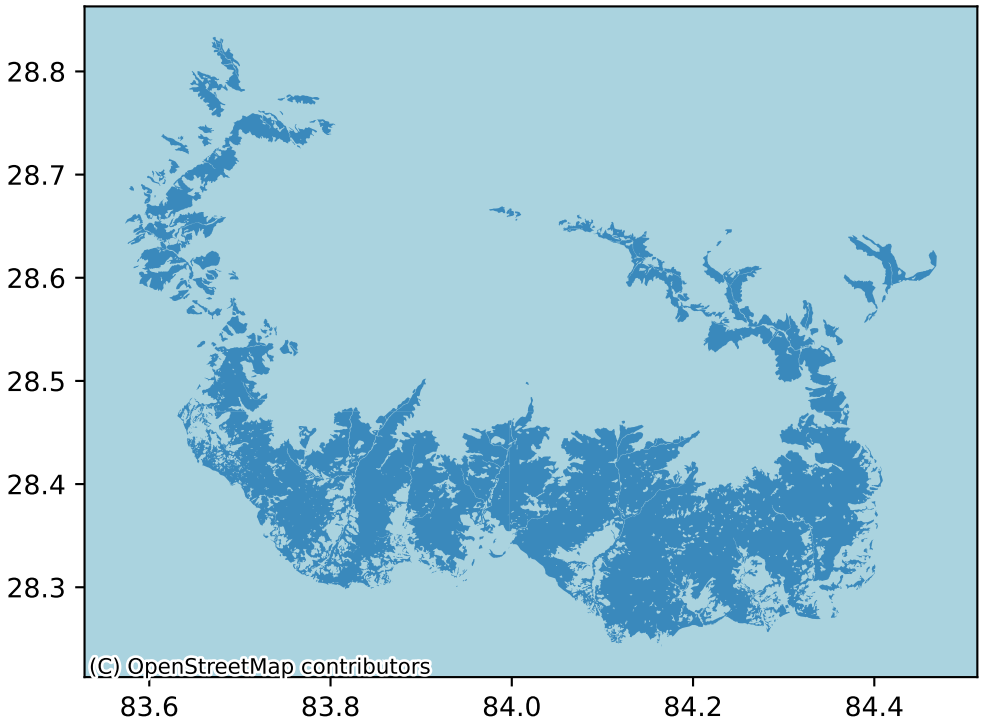
```r
library(ggplot2)
library(ggspatial)

ggplot() +
  annotation_map_tile(zoomin = -1) +
  geom_sf(data = forest, fill = "green", color = "darkgreen") +

  labs(title = "Forest Areas in Nepal") +
  theme_minimal()
```

```
Zoom: 9
```



Python

```
import matplotlib.pyplot as plt
import contextily as ctx

fig, my_map = plt.subplots(figsize=(6, 6))

forest.plot(ax=my_map,linewidth=0.03, scheme="Quantiles", k=5, alpha=0.8, legend=False)

ctx.add_basemap(my_map, source = ctx.providers.OpenStreetMap.Mapnik)
```



Loading MODIS data from planetary computer

We are going to load MODIS FPAR data from planetary computer as the indicator that we can connect with every polygon. To find the relevant data we use the Spatio Temporal Asset Catalog (STAC) to search through the data available at the Planetary Computer. Here we filter the data with the bounding box of our AOI we constructed earlier and we select the data for the August 2023. Once we found the list of relevant datasets we can download them to our local computer. We get a lazy representation of the relevant datasets on the planetary computer.

> **Note**
>
> Although we use the same date-time filters in rstac (in R) and pystac (in Python) of August 1st to 31st 2023, the items retrieved by both libraries are not the same, as they each parse time differently.

R

```r
# Download
library(rstac)

# Define the Planetary Computer STAC endpoint
planetary_computer_stac_url <- "https://planetarycomputer.microsoft.com/api/stac/v1"

# Connect to the Planetary Computer STAC API
stac_source <- stac(planetary_computer_stac_url)

stac_query <- rstac::stac_search(
  q = stac_source,
  collections = "modis-15A2H-061",
  bbox = nepal_bbox,
  datetime = "2023-08-01/2023-08-31"
)

executed_stac_query <- rstac::get_request(stac_query);
executed_stac_query
```

Python

```python
catalog = pystac_client.Client.open(
    "https://planetarycomputer.microsoft.com/api/stac/v1",
    modifier=planetary_computer.sign_inplace,
)

search = catalog.search(
    collections="modis-15A2H-061",
    datetime="2023-08-01/2023-08-30",
    bbox=bbox
)
```

```
items = list(search.items())
#print(f"Found {len(items)} items:")
items
```

```
[<Item id=MYD15A2H.A2023241.h25v06.061.2023251215447>, <Item id=MOD15A2H.A2023241.h25v06.061
```

Now we download the selected data to our computer. Here we can select the assets of the dataset that we want to download. In our case this is the 500m resolution FPAR and its metadata.

R

```
signed_stac_query <- rstac::items_sign(
  executed_stac_query,
  rstac::sign_planetary_computer()
)

signed_stac_query

rstac::assets_download(signed_stac_query, c("Fpar_500m", "metadata"), output_dir = tempdir()
output_file <- file.path(
  tempdir(),
  "modis-061-cogs"
  )

cog_files <- list.files(path = output_file,
                        pattern = "\\.tif$",
                        full.names = TRUE,
                        recursive = TRUE)
cog_files
```

Python

```
import odc.stac
data = odc.stac.load(
    items,
    crs="EPSG:3857",
    bands="Fpar_500m",
    resolution=500,
```

```
    bbox=bbox,
)

raster = items[2].assets["Fpar_500m"].extra_fields["raster:bands"]
data = data * raster[0]["scale"]
data
```

```
<xarray.Dataset> Size: 2MB
Dimensions:      (y: 278, x: 222, time: 5)
Coordinates:
  * y            (y) float64 2kB 3.418e+06 3.417e+06 ... 3.28e+06 3.279e+06
  * x            (x) float64 2kB 9.293e+06 9.293e+06 ... 9.403e+06 9.403e+06
    spatial_ref  int32 4B 3857
  * time         (time) datetime64[ns] 40B 2023-07-28 2023-08-05 ... 2023-08-29
Data variables:
    Fpar_500m    (time, y, x) float64 2MB 0.2 0.06 2.53 2.53 ... 0.9 0.92 0.92
```

Compute Zonal statistics

We are converting the forest polygons and the raster data of the indicator into a vector data cube by computing the zonal statistics of the MODIS FPAR for every polygon. We compute the minimum, mean, and maximum in space to get three timeseries for every polygon.

R

```
library(stars)
```

```
Loading required package: abind
```

```
stars::read_stars(cog_files, along = 'time', proxy=FALSE) |>
  st_transform(st_crs(forest)) -> np_data

forest_outline = st_make_valid(forest)

vecData_mean <-  aggregate(np_data, by = forest_outline, FUN = mean)
vecData_max  <- aggregate(np_data, by = forest_outline, FUN = max)
vecData_min <- aggregate(np_data, by = forest_outline, FUN = min)
```

Python

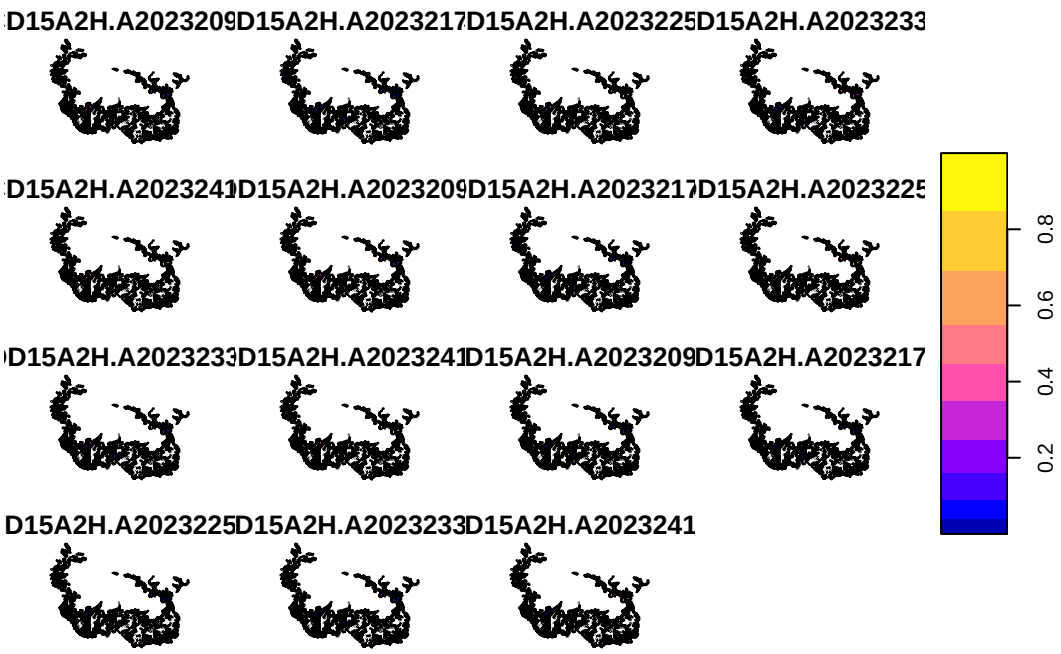
```
import xvec
forest = forest.to_crs(epsg=3857)

#aggregated = data.xvec.zonal_stats(
#    forest.geometry,
#    x_coords="x",
#    y_coords="y",
#    stats=[
#        "mean",
#        "min",
#        "max"
#    ],
#)
#aggregated
```

Plot the Zonal statistics

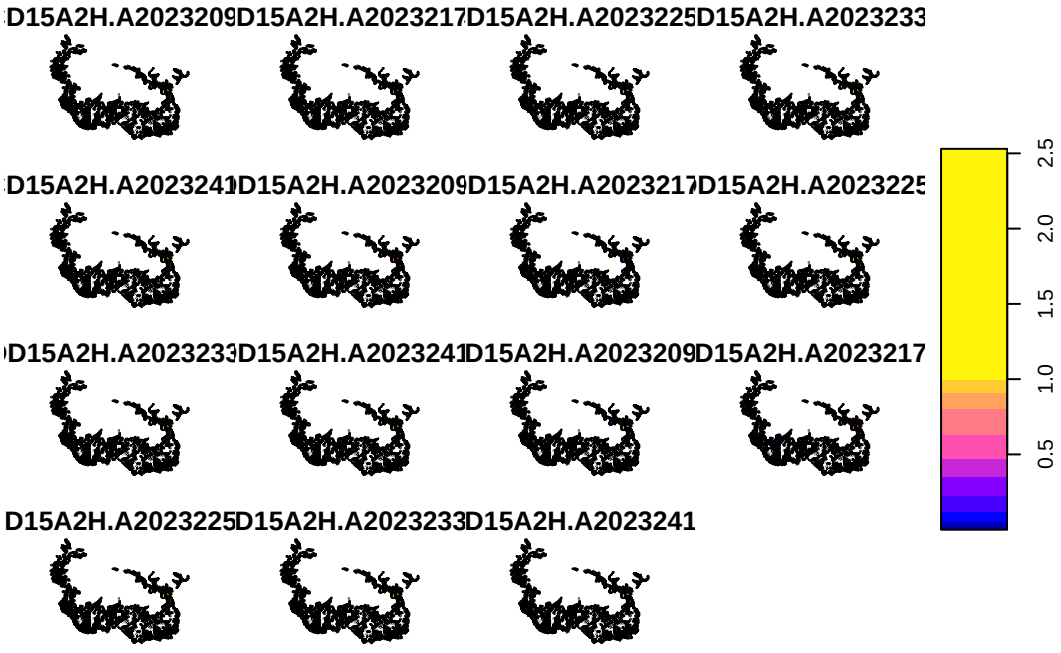
In R, using the library stars, we can plot the AOI polygons with their computed zonal statistics of the MODIS FPAR. This feature is still under development in Python in the xvec library. Therefore we are only filtering the data for a specific forest district.

R

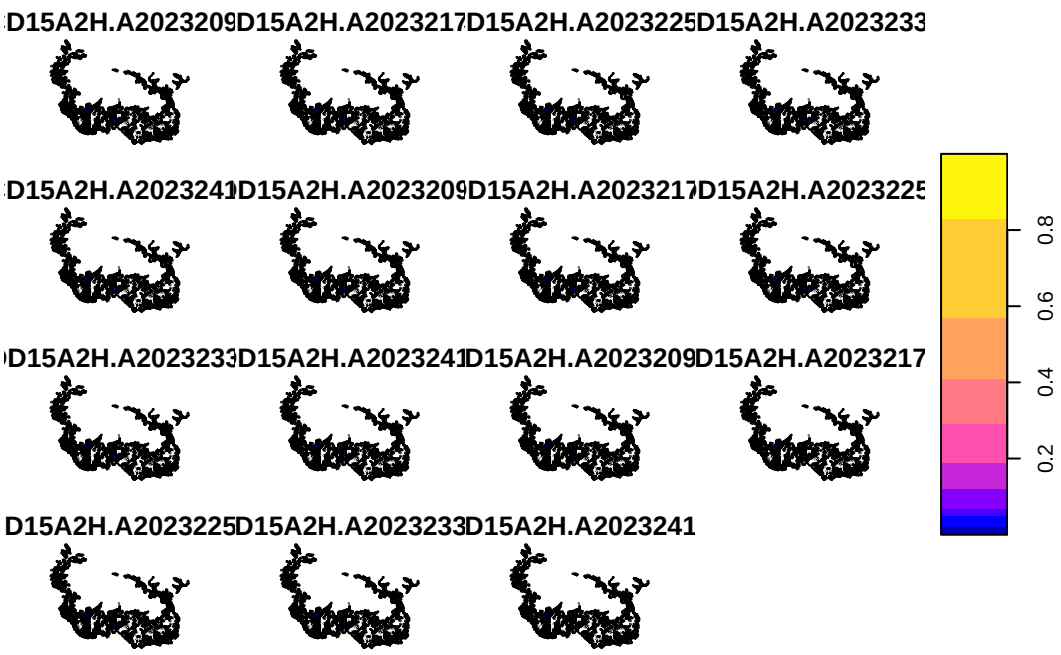
```
vecData_mean |>
  slice(index = 1:15, along = "time") |>
  plot(downsample = c(3, 3, 1), max.plot = 15)
```




```
vecData_max |>
  slice(index = 1:15, along = "time") |>
  plot(downsample = c(3, 3, 1), max.plot = 15)
```



```r
vecData_min |>
  slice(index = 1:15, along = "time") |>
  plot(downsample = c(3, 3, 1), max.plot = 15)
```



Python

```python
forest_id = 789

# Filter the GeoDataFrame to get the row corresponding to the specific district
forest_row = forest[forest['fid'] == forest_id]

# Extract the geometry of the specific district
forest_geometry = forest_row.geometry.iloc[0]
forest_geometry
```

```
<POLYGON ((9314972.393 3354069.165, 9314983.662 3354080.662, 9314997.869 335...>
```

```python
#aggregated.sel(geometry=forest_geometry)
```

Summary

In this tutorial you learned how to open a shapefile and a raster dataset to create a vector data cube of agreggated values, in our case zonal statistics of the raster data for the given forest polygons. This vector data cube can then be used for further analysis. We also learned how to use STAC to select and retrieve data from a cloud provider.