

How to convert NetCDF to Points/ Polygons

What is NetCDF?

Dong-Hoon, Lee (이동훈)
dhl@gaia3d.com

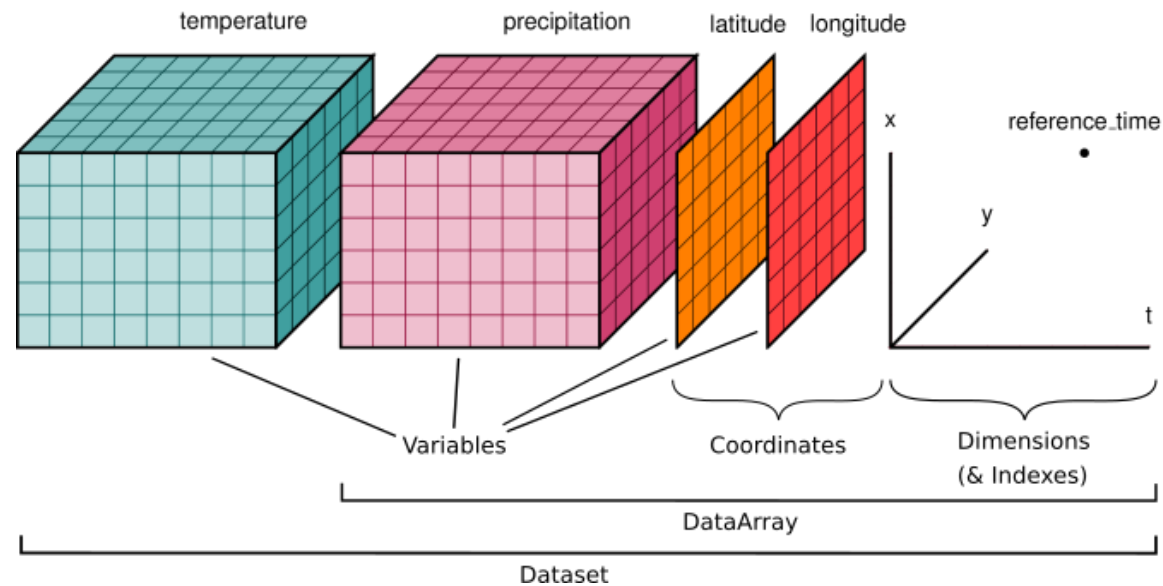
What is NetCDF?

- NetCDF (Network Common Data Form) is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data
- It is commonly used in climatology, meteorology and oceanography applications (e.g., weather forecasting, climate change) and GIS applications.
- <https://en.wikipedia.org/wiki/NetCDF>

Climatology & Oceanography require time-series and three-dimensional measured information, NetCDF is so useful.

Multidimensional spatio-temporal data

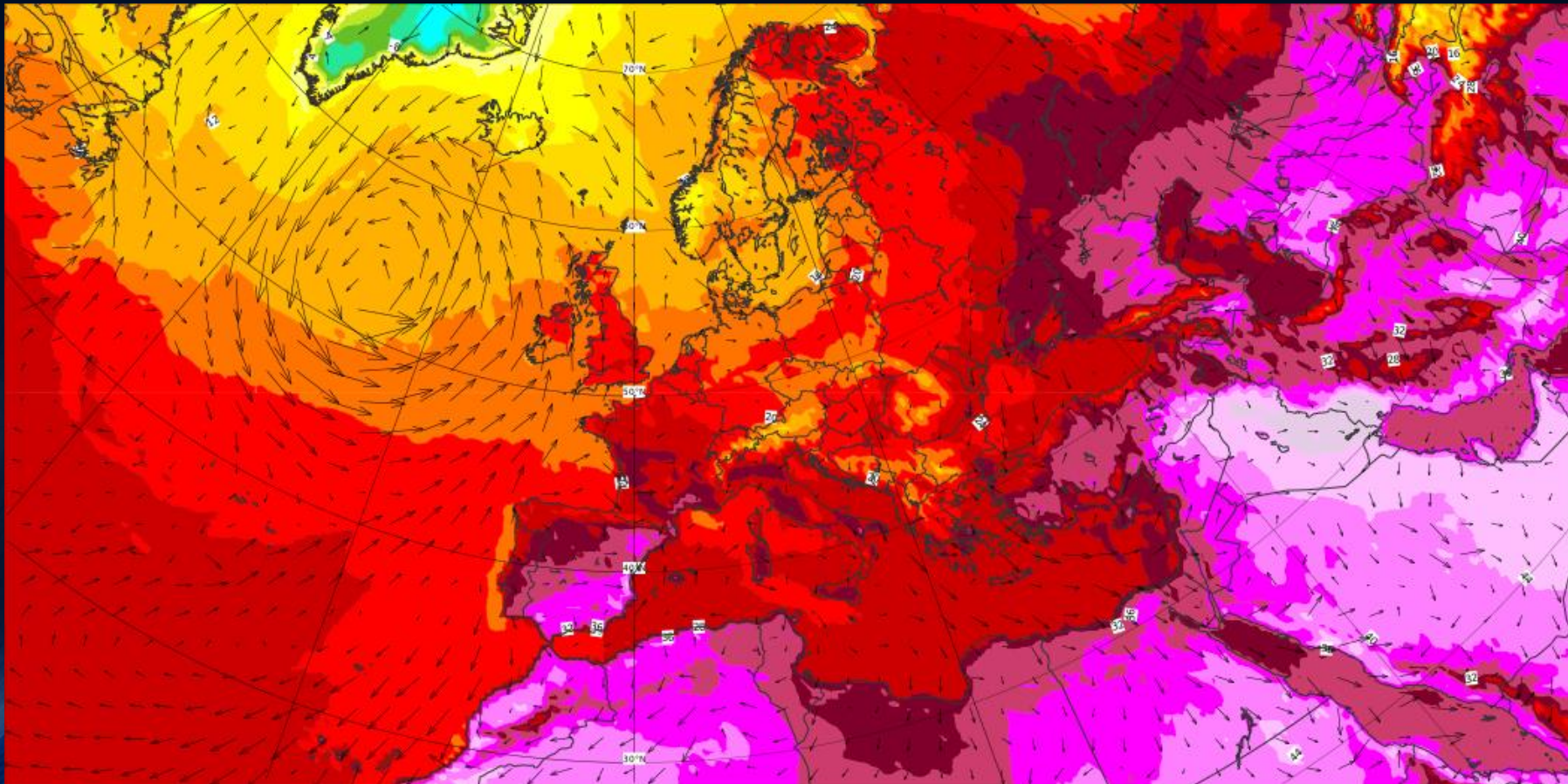
- NetCDF can store multidimensional spatio-temporal information
- Xarray is a Python package that can use multidimensional spatio-temporal data including NetCDF



<https://docs.xarray.dev/en/stable/user-guide/data-structures.html>

Where can you get that Data?

- ECMWF (<https://www.ecmwf.int/>)
 - ECMWF is the European Centre for Medium-Range Weather Forecasts



Copernicus (<https://www.copernicus.eu/>)



ATMOSPHERE	EMERGENCY - CEMS-EFFIS	LAND - PAN-EUROPEAN/LOCAL LAND
CLIMATE CHANGE	EMERGENCY - CEMS-GLOFAS	MARINE
EMERGENCY - CEMS-EFAS	LAND - GLOBAL	

Multi Observation Global Ocean 3D Temperature Salinity Height Geostrophic Current and MLD



Home > Marine Data Store > Product

i	Description
🔔	Notifications
📄	Data access
✉	Contacts
DOCUMENTATION	
📖	User Manual
📄	Quality Information Document
📄	Licence
📄	How to cite
DOI	
🏠	10.48670/moi-00052

Data access and mapping services

There are multiple ways to download data from this product:

- If you prefer a graphical tool, click on above.
- To subset data in time and/or space, choose MOTU.
- If you use an OPeNDAP client such as *netCDF4/xarray* (Python), *ferret*, or *MATLAB*, choose OPeNDAP or ERDDAP.
- **Raw files:** Use FTP or the web-based File Browser.
- **Maps:** To request maps from QGIS or similar tools, use our Web Mapping Service (WMS).

Dataset	MOTU i	OPeNDAP i	ERDDAP i	Raw files i	Maps i
dataset-armor-3d-nrt-weekly	Link	Link	-	FTP Browse	WMS
dataset-armor-3d-nrt-monthly	Link	Link	-	FTP Browse	WMS
dataset-armor-3d-rep-weekly	Link	Link	-	FTP Browse	WMS
dataset-armor-3d-rep-monthly	Link	Link	-	FTP Browse	WMS

https://data.marine.copernicus.eu/product/MULTIOBS_GLO_PHY_TSUV_3D_MYNRT_015_012/services

LDAPS and ETC...

- <https://data.kma.go.kr/data/rmt/rmtList.do?code=340&pgmNo=65>

기상청 기상자료개방포털

국가기후데이터센터 소개 | *가-가 | 로그인 | 사이트맵 | ☆ 즐겨찾기 | ENG(info)

'관측'을 검색하세요

인기검색어

기상자료개방포털이란? 데이터 기후통계분석 간행물 소통과 참여

데이터

국지예보모델(LDAPS)

국지예보모델(LDAPS) - 파일셋

■ 자료설명

국지예보모델(UM 1.5kmL70)의 공간 해상도는 1.5km이며, 연직으로 약 40km까지 70층으로 구성되며, 3시간 간격으로 전지구모델로부터 경계장을 제공받아 1일 8회(00, 06, 12, 18UTC : 48시간 예측, 03, 09, 15, 21UTC : 3시간 예측) 수행합니다.

국지예보모델은 3차원 변분자료 동화 기법을 이용하여 각각의 자체 분석-예측 순환 체계로 운영하고 있으며, 국지예보모델의 산출자료는 등압면, 모델면, 단일면 자료 3종류가 제공되며, 데이터 형식은 WMO에서 제시한 GRIB2 형식으로 제공합니다.

< 자료요청 방법 >

1. 검색을 통해 조회된 결과 목록에서 다운로드 받을 자료 선택 후 담기 버튼 선택합니다.
2. '마이페이지 > 대용량자료신청대기목록'에서 다운로드를 신청합니다.

※ 최대 신청 가능한 용량은 100GB입니다.

< 경량화 파일 >

단일면 주요 기상요소 20종에 대한 파일셋 구성하여 '경량화' 탭 통해 제공(제공기간/방법: 2021.1.~ / 웹 다운로드)

* 1회 최대 신청용량 100GB

* 최근 1년 자료까지만 제공

기상관측

기상위성

레이더

기상예보

수치모델

수치분석일기도

단·중기예측

- 지역예보모델(RDAPS)

- 국지예보모델(LDAPS)

GRIB (GRIdded Binary or General Regularly-distributed Information in Binary form[1]) is a concise data format commonly used in meteorology to store historical and forecast weather data. It is standardized by the World Meteorological Organization's Commission for Basic Systems, known under number GRIB FM 92-IX, described in WMO Manual on Codes No.306

<https://en.wikipedia.org/wiki/GRIB>

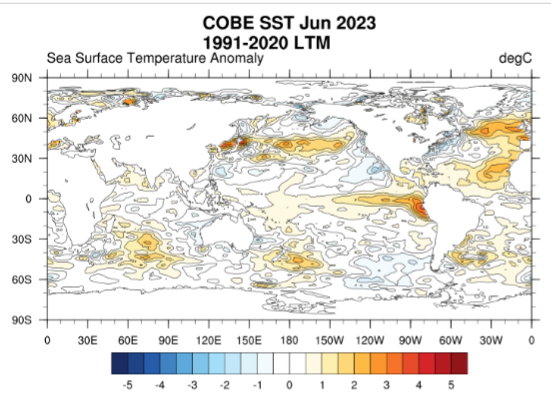


- <https://psl.noaa.gov/data/gridded/data.cobe.html>

Home » Data » Gridded Climate » COBE Sea Surface Temperature

COBE Sea Surface Temperature

Monthly 1° x 1° SST dataset from 1891 to present from the JMA. Datasets uses 3D Var to fill gaps.



Download and Plot Data [Data Help](#)

Search Dataset Variables (start with * to match anywhere) [Clear](#)

+ / -

Variable	Statistic	Level	TimeScale	Options
Sea Surface Temperature ▾				
2 Sea Surface Temperature	Long Term Mean	Surface	Monthly	Download Plot
1 Sea Surface Temperature	Mean	Surface	Monthly	Download Plot

[/Datasets/COBE/sst.mon.mean.nc](#) [Download](#) [Plot](#)

Thredds Catalog: [Download](#) [Plot/Subset](#)

SPECIFICATIONS

Temporal Coverage

- Monthly Means: 1891/01 to 2023/06
- Long Term Mean: 1991-2020 (default; 1981-2010 also available)

Spatial Coverage

- 1.0° latitude x 1.0° longitude global grid (360x180)
- 89.5N - 89.5S, 0.5E - 359.5E

Levels:

- Sea Level

Update Schedule:

- Monthly

DESCRIPTION

USAGE & CITATION

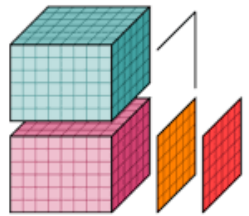
SOURCE & REFERENCES

VARIABLES

[sst.mon.mean.nc](#)

160,775KB

Data Processing Environment



xarray



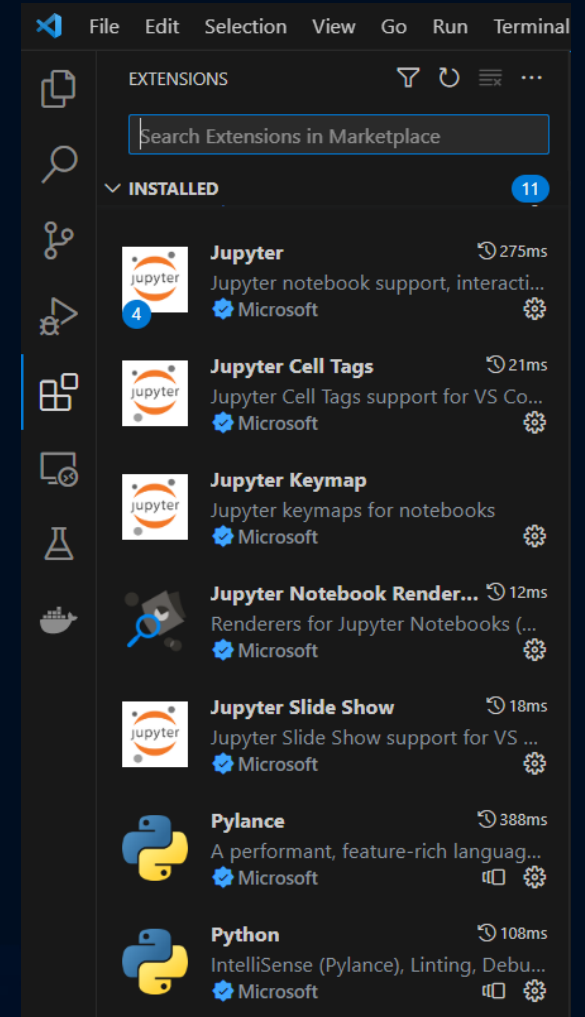
GeoPandas



python



Visual Studio Code



Loading NetCDF by using Xarray

```
import xarray as xr #netcdf와 같은 다차원 배열 데이터 처리용 패키지
```

✓ 0.0s







```
ds = xr.open_dataset('sst.mon.mean.nc')  
ds
```

✓ 0.0s


xarray.Dataset

– Dimensions: (lat: 180, lon: 360, time: 1590)

▼ Coordinates:

lat	(lat)	float32	89.5 88.5 87.5 ... -88.5 -89.5	 
lon	(lon)	float32	0.5 1.5 2.5 ... 357.5 358.5 359.5	 
time	(time)	datetime64[ns]	1891-01-01 ... 2023-06-01	 

▼ Data variables:

sst	(time, lat, lon)	float32	...	 
-----	------------------	---------	-----	---

– Indexes: (3)

▼ Attributes:

title : COBE Sea Surface Temperature Analysis
history : created June 2011 at ESRL/PSD using the grib data from JRA.
platform : Analyses
original_source : http://near-goos1.jodc.go.jp/cgi-bin/1997/near_goos_catalog?projectname=NEA
R-GOOS
Conventions : CF-1.2
institution : NOAA ESRL/PSD
comment : recent values (w/i last year) may change as dataset is updated. It is a monitoring dataset.
dataset_title : COBE Sea Surface Temperature
References : <https://www.psl.noaa.gov/data/gridded/data.cobe.html>

- import xarray package

- loading data & View

Dimension

Z(Depth/Height) Dimension
also can be added

Variables – Sea Surface Temperature

Metadata

Extract data by temporal ranges

```
# import package for deal with datetime data
import datetime
```

✓ 0.0s

```
# extract the data by time - recent 2 years
time_start = datetime.datetime(2022, 1, 1, 0, 0, 0)
time_end = datetime.datetime(2022, 12, 31, 23, 59, 59)
ds2 = ds.sel(time=slice(time_start, time_end))
ds2
```







✓ 0.0s

dataset.sel(time index=slice(start, end))



xarray.Dataset

▸ Dimensions: (lat: 180, lon: 360, time: 12)

▼ Coordinates:

lat	(lat)	float32	89.5 88.5 87.5 ... -88.5 -89.5	 
lon	(lon)	float32	0.5 1.5 2.5 ... 357.5 358.5 359.5	 
time	(time)	datetime64[ns]	2022-01-01 ... 2022-12-01	 

▼ Data variables:

sst	(time, lat, lon)	float32	...	 
-----	------------------	---------	-----	---

▸ Indexes: (3)

▸ Attributes: (9)

Extract data by spatial ranges

```
# extract the data by spatial - zoom in asia
min_lon = 120
min_lat = 30
max_lon = 140
max_lat = 45
# mask the coordinates
mask_lon = (ds2.lon >= min_lon) & (ds2.lon <= max_lon)
mask_lat = (ds2.lat >= min_lat) & (ds2.lat <= max_lat)
# clip the dataset
ds3 = ds2.where(mask_lon & mask_lat, drop=True)
ds3
```







dataset.where(coordinates range)

✓ 0.0s


xarray.Dataset

▸ Dimensions: (time: 12, lat: 15, lon: 20)

▼ Coordinates:

lat	(lat)	float32	44.5 43.5 42.5 ... 32.5 31.5 30.5	 
lon	(lon)	float32	120.5 121.5 122.5 ... 138.5 139.5	 
time	(time)	datetime64[ns]	2022-01-01 ... 2022-12-01	 

▼ Data variables:

sst	(time, lat, lon)	float32	nan nan nan ... 21.97 22.07 22.16	 
-----	------------------	---------	-----------------------------------	---

▸ Indexes: (3)

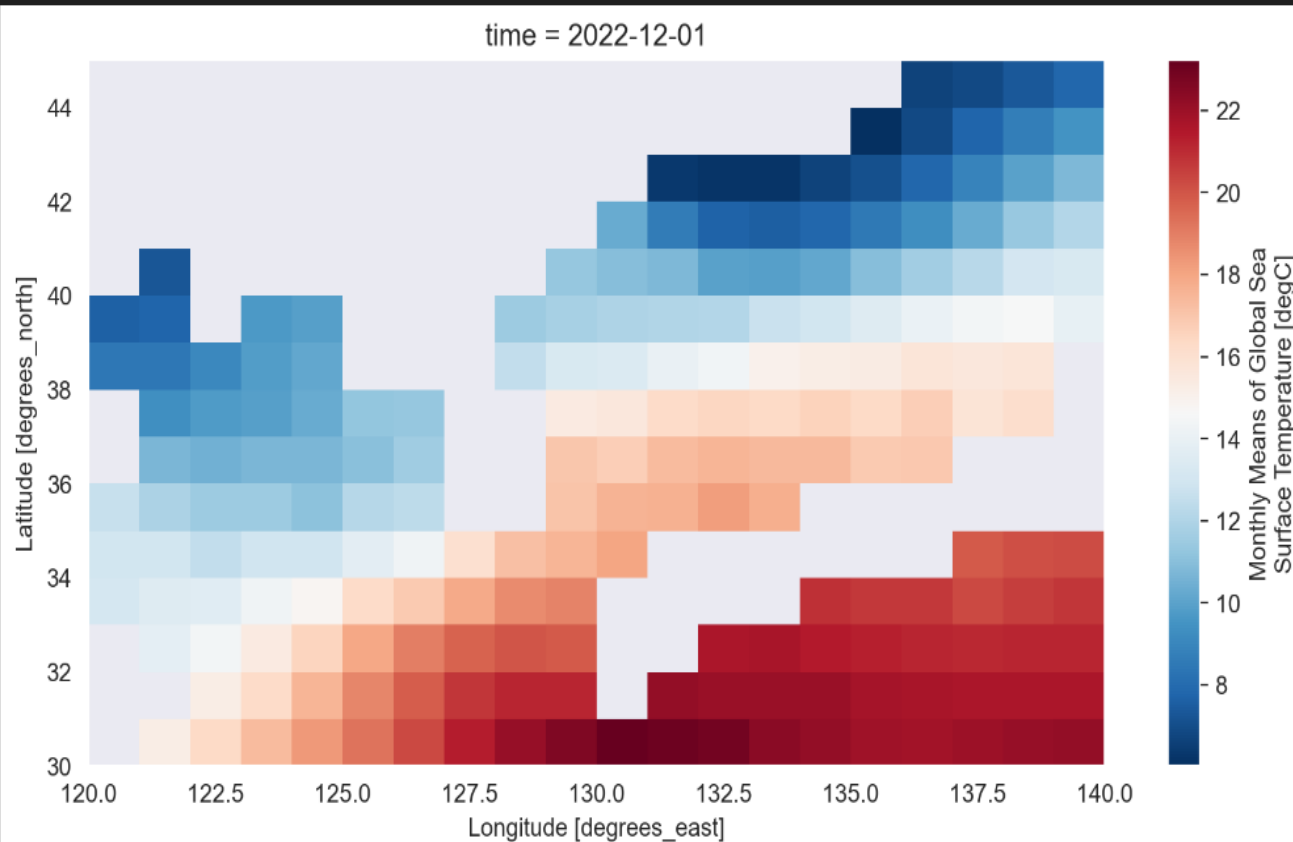
▸ Attributes: (9)

Plot the snapshot of dataset

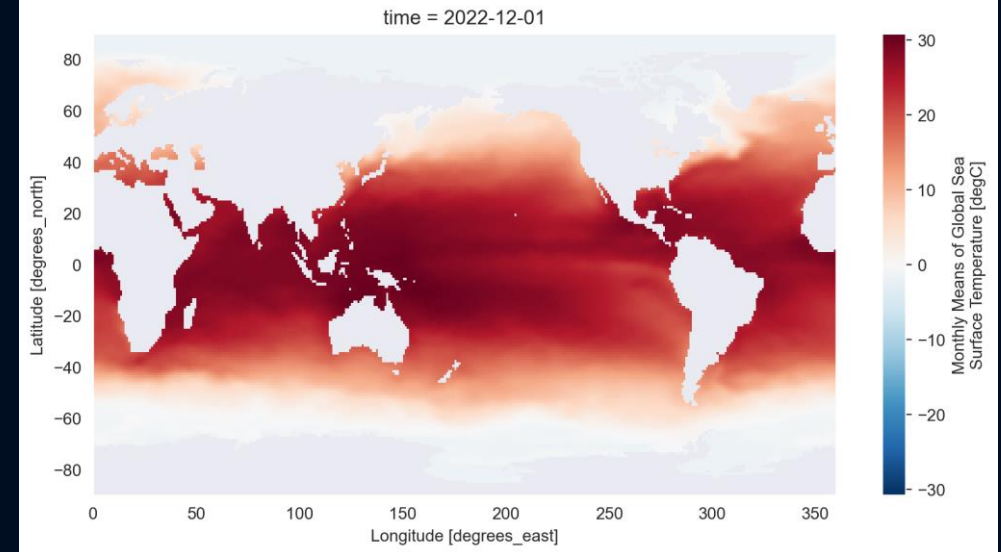
```
# you can plot the snapshot of dataset
snapshot = datetime.datetime(2022, 12, 1, 0, 0, 0)
ds3.sel(time = snapshot).sst.plot(cmap='RdBu_r',figsize=(10,5))
```

✓ 0.5s

<matplotlib.collections.QuadMesh at 0x1e888380d10>



```
# you can plot the snapshot of dataset
snapshot = datetime.datetime(2022, 12, 1, 0, 0, 0)
ds.sel(time = snapshot).sst.plot(cmap='RdBu_r',figsize=(10,5))
```



Now, convert NetCDF into Pandas DataFrame

- using Xarray's to_dataframe

```
# Latitude, longitude and time are indexed.  
# You can convert it to a general attribute column by using reset_index.  
df = ds3.to_dataframe().reset_index()  
df.info()  
df.tail()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3600 entries, 0 to 3599  
Data columns (total 4 columns):  
#   Column  Non-Null Count  Dtype  
---  ---  
0    time    3600 non-null    datetime64[ns]  
1    lat      3600 non-null    float32  
2    lon      3600 non-null    float32  
3    sst      2364 non-null    float32  
dtypes: datetime64[ns](1), float32(3)  
memory usage: 70.4 KB
```

	time	lat	lon	sst
3595	2022-12-01	30.5	135.5	21.897499
3596	2022-12-01	30.5	136.5	21.852501
3597	2022-12-01	30.5	137.5	21.965000
3598	2022-12-01	30.5	138.5	22.072500
3599	2022-12-01	30.5	139.5	22.162500

Make Point GeoDataFrame from lat, lon coordinates

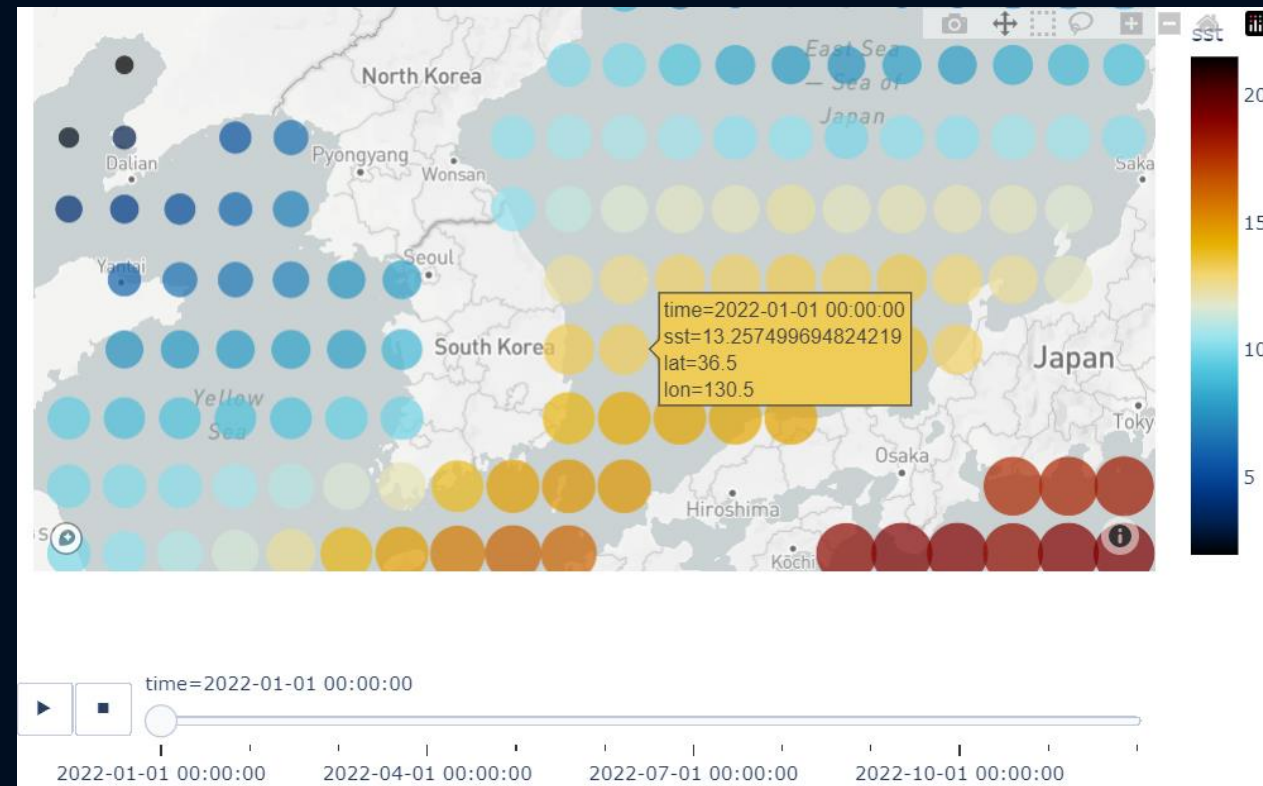
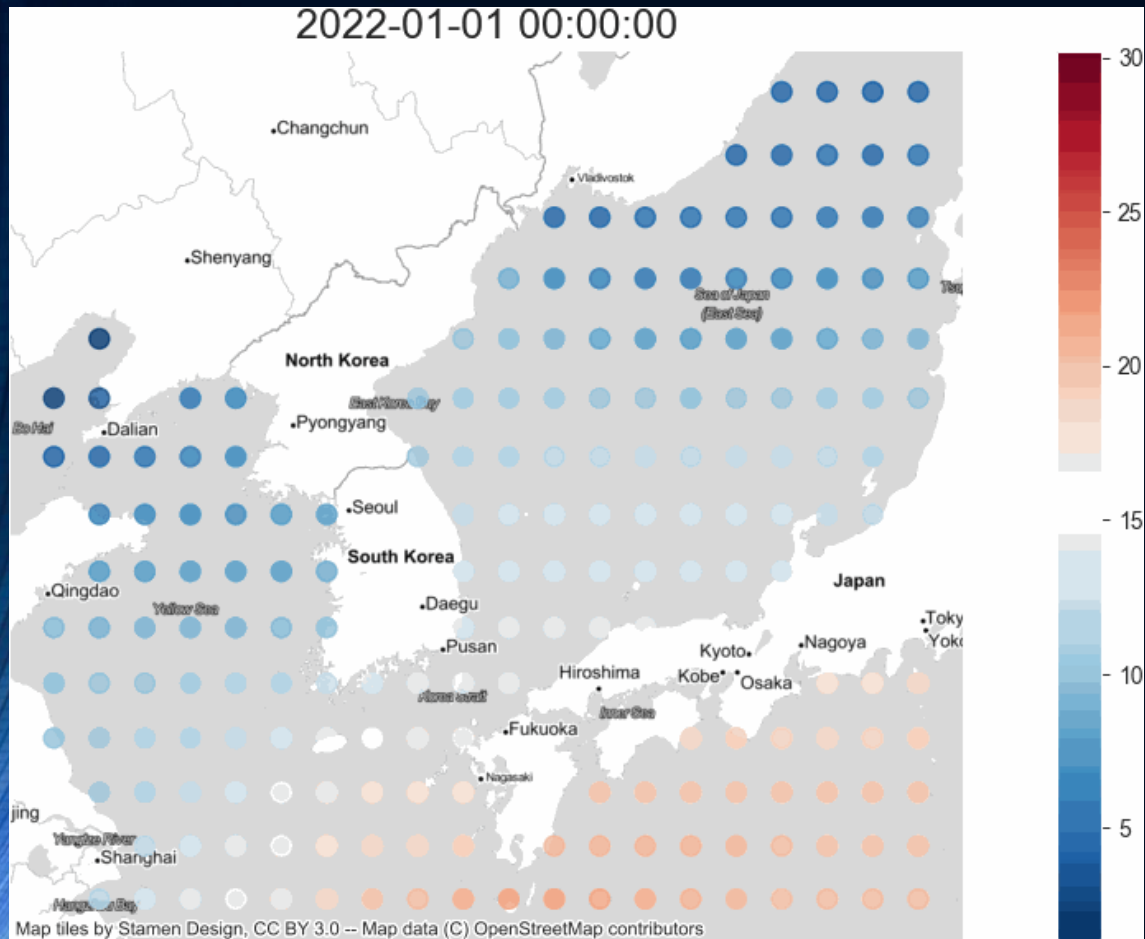
```
# Make Point GeoDataFrame from lat, lon coordinates
# make point geometry
geom = gpd.points_from_xy(df_nnull.lon, df_nnull.lat)
gdf_pt4326 = gpd.GeoDataFrame(df_nnull, geometry=geom, crs='EPSG:4326')
gdf_pt4326.info()
gdf_pt4326.head()
```

✓ 0.0s

```
<class 'geopandas.geodataframe.GeoDataFrame'>
Index: 2364 entries, 16 to 3599
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   time        2364 non-null   datetime64[ns]
1   lat         2364 non-null   float32
2   lon         2364 non-null   float32
3   sst         2364 non-null   float64
4   geometry    2364 non-null   geometry
dtypes: datetime64[ns](1), float32(2), float64(1), geometry(1)
memory usage: 92.3 KB
```

	time	lat	lon	sst	geometry
16	2022-01-01	44.5	136.5	4.140001	POINT (136.50000 44.50000)
17	2022-01-01	44.5	137.5	3.992501	POINT (137.50000 44.50000)
18	2022-01-01	44.5	138.5	3.752501	POINT (138.50000 44.50000)
19	2022-01-01	44.5	139.5	4.035001	POINT (139.50000 44.50000)
35	2022-01-01	43.5	135.5	4.190001	POINT (135.50000 43.50000)

Make Map with Point (Time Animation)



Make Polygon (Grid) Data

```
# user function for make grid ploygon by 1 degree
def create_polygon(lat, lon):
    return Polygon([(lon, lat),
                    (lon + 1, lat),
                    (lon + 1, lat + 1),
                    (lon, lat + 1)])
```

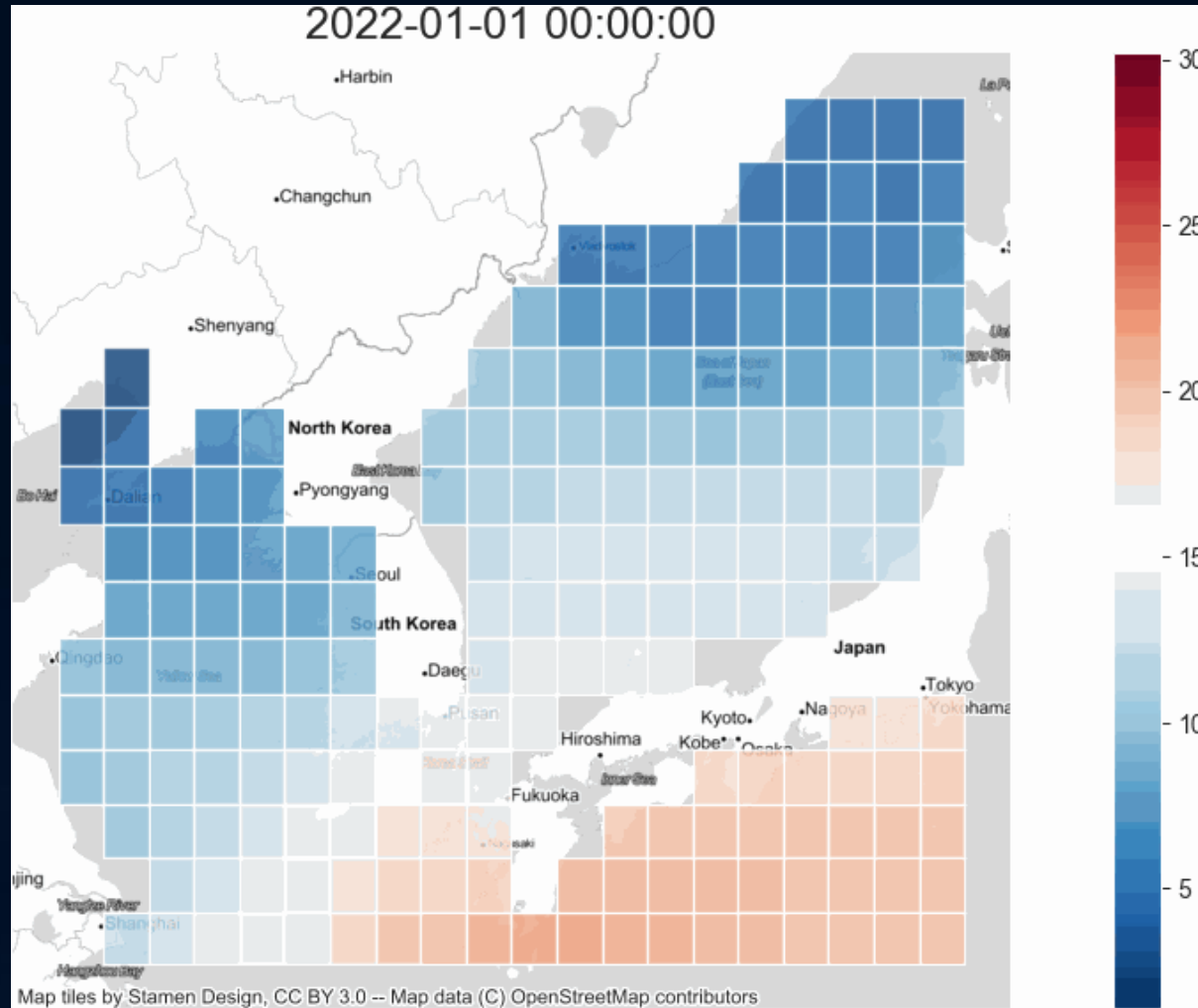
✓ 0.0s

```
# Make polygon geometry with "create_polygon" function
geom_pg = df.apply(lambda row: create_polygon(row['lat'], row['lon']), axis=1)

# Make Polygon GeoDataFrame with DataFrame & Polygon geometry
gdf_pg4326 = gpd.GeoDataFrame(df, geometry=geom_pg, crs='4326')
```

✓ 0.2s



Make Map with Polygon Grid



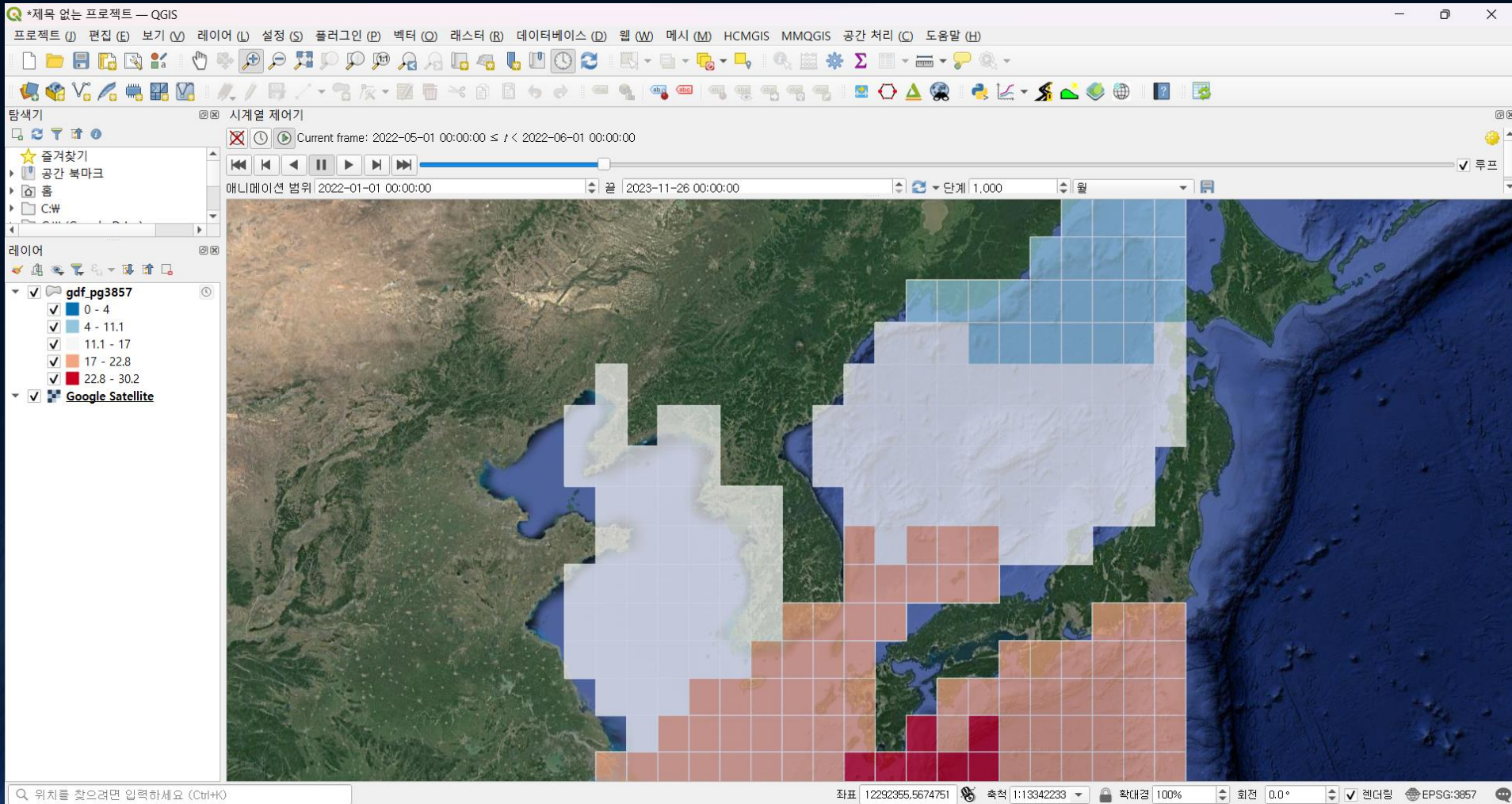
Save GIS File

- Using GeoPandas

```
# save shp file
gdf_pt3857.to_file('shp/gdf_pt3857.gpkg', driver="GPKG")
gdf_pg3857.to_file('shp/gdf_pg3857.gpkg', driver="GPKG")
✓ 1.1s
```

 gdf_pg3857.gpkg	944KB
 gdf_pt3857.gpkg	416KB

Using GeoPackage in QGIS or ETC



Thank You

Dong-Hoon, Lee (이동훈)
dhl@gaia3d.com