JAXA Earth API for Python及びQGISを使用した地球観測衛星画像の利用

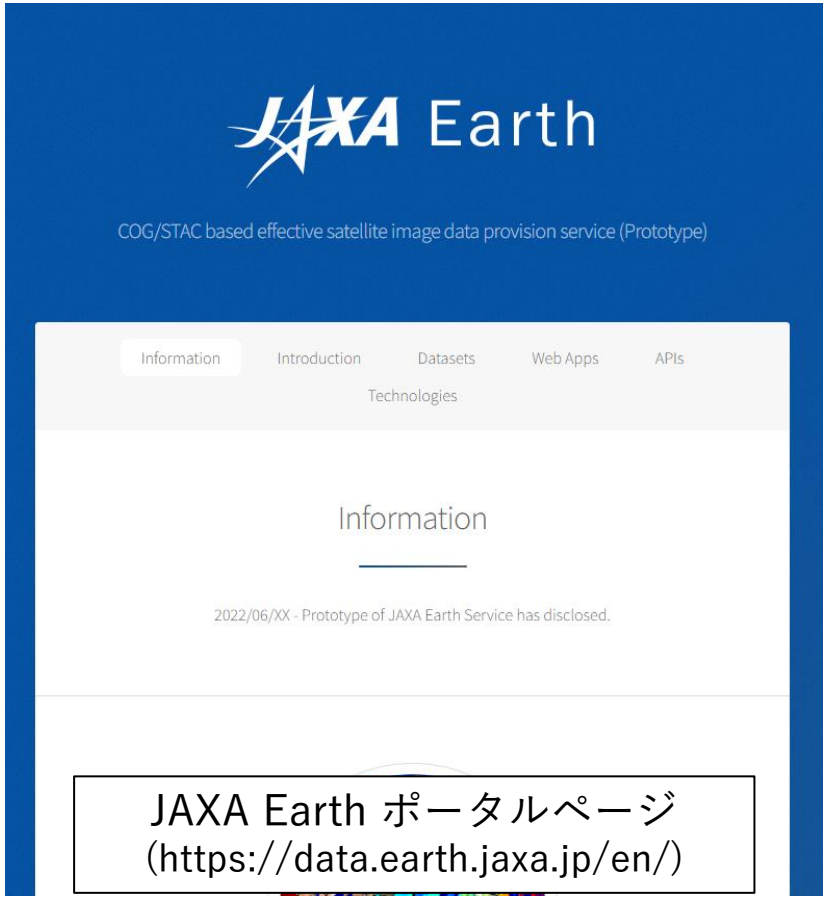
JAXA 第一宇宙技術部門 地球観測研究センター(EORC)

2022年6月

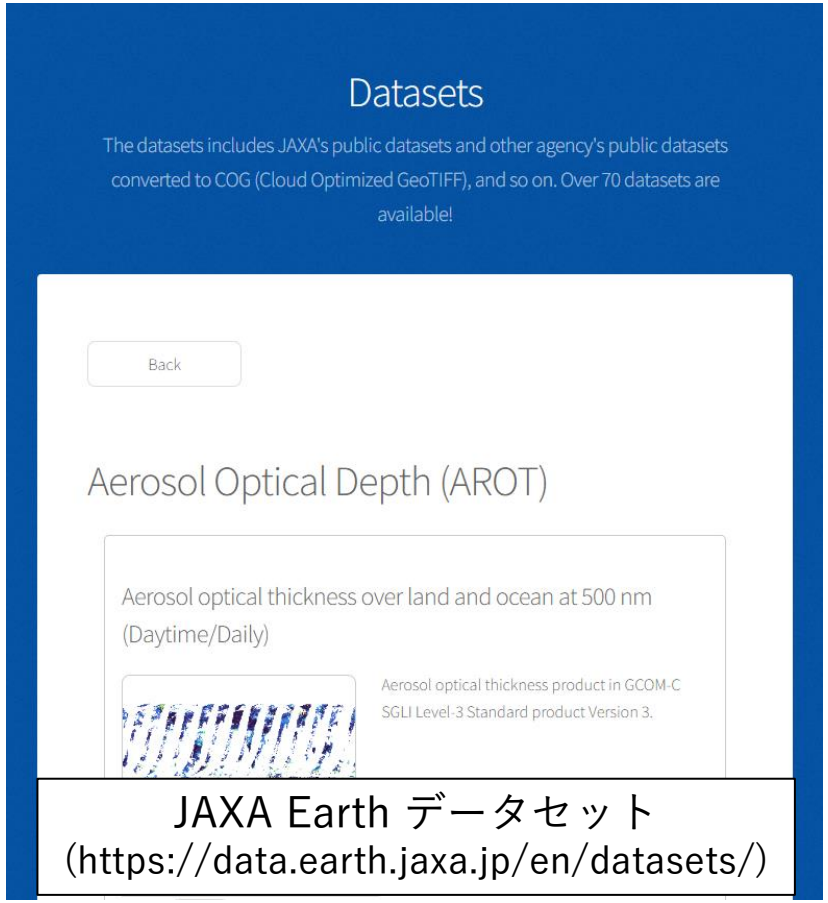
内容

- JAXA Earth API for Python 概要
- 事前準備、基本的な使用方法
- データ利用と簡単な分析事例
- まとめ、今後の予定

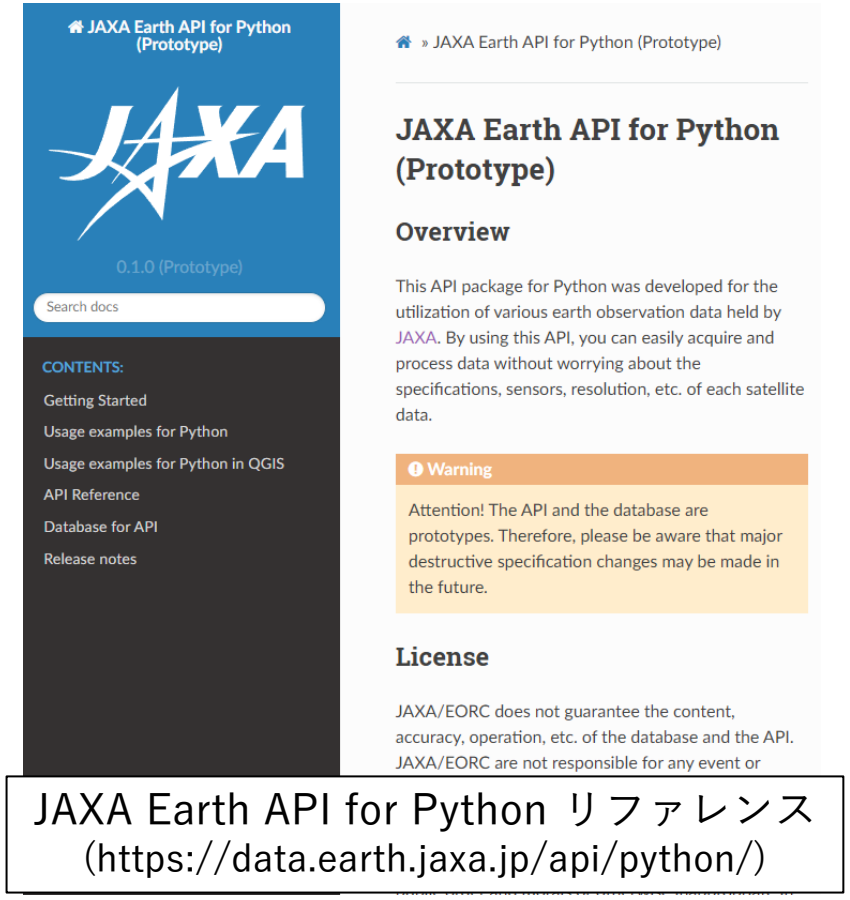
JAXA Earth API for Python：概要
- ホームページ -



JAXA Earth ポータルページ
(https://data.earth.jaxa.jp/en/)



JAXA Earth データセット
(https://data.earth.jaxa.jp/en/datasets/)



JAXA Earth API for Python リファレンス
(https://data.earth.jaxa.jp/api/python/)

JAXA Earthサービス(Webアプリ・データベース・API)は2022年6月9日より公開開始。
API本体、インストール方法、使用例、詳しい説明等はリファレンスページ（英語）に記載。

JAXA Earth API for Python：概要
- 概要及び機能 -

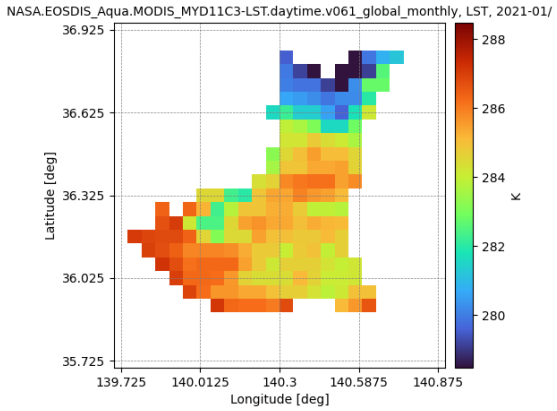
概要

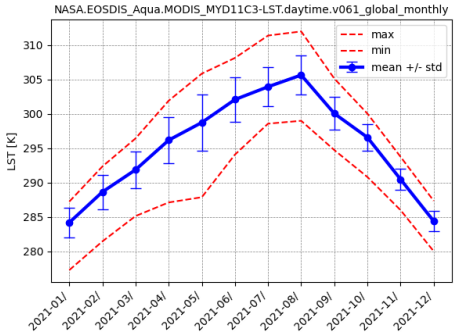
・ユーザーリクエストに応じた衛星画像、時系列グラフ等をPythonの短いプログラミングで容易に取得、可視化が可能なクライアント型API.

機能

・プロダクト名検索機能
・リクエストに応じた衛星画像の自動取得機能
(プロダクト、分解能、日付範囲、関心領域等)
・マスク処理、時空間統計処理機能
・geojsonによる関心領域(行政界等)マスク機能
・カラーマップ、カラーレンジ設定機能
・QGISへのインターフェース機能

```python
# Load module
from jaxa.earth import je

# Set Parameters
key  = ["ndvi","half","normal"]
date = ["2021-01-01T00:00:00",
        "2021-12-31T00:00:00"]
ppu  = 90
bbox = [137.5, 33.0, 142.5, 38.0]

# Aquire information of data, image
collections,bands = je.ImageCollectionList()\
                      .filter_name(keywords=key)
data = je.ImageCollection(collection=collections[0])\
         .filter_date(date=date)\
         .filter_resolution(ppu=ppu)\
         .filter_bounds(bbox=bbox)\
         .select(band=bands[0][0])\
         .get_images()

# Processing and showing images
img = je.ImageProcess(data)\
        .show_images()
```

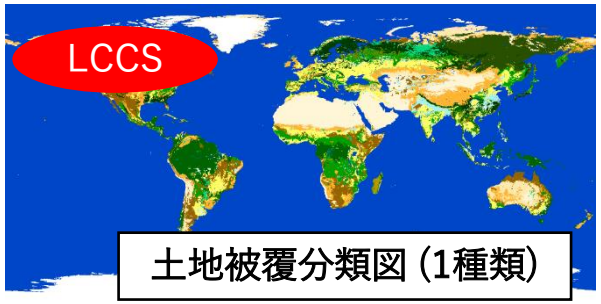




JAXA Earth API for Python 呼出、実行結果例

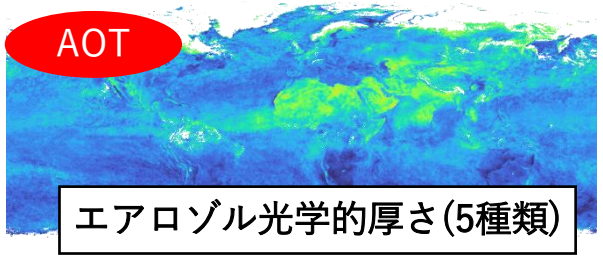
QGISにてAPI for Pythonを使用することで、地球環境データの取得、処理が可能。なお、Google colab/jupyter notebook等で動作させることも可能。

JAXA Earth API for Python：概要
- 利用可能な地球環境データ -



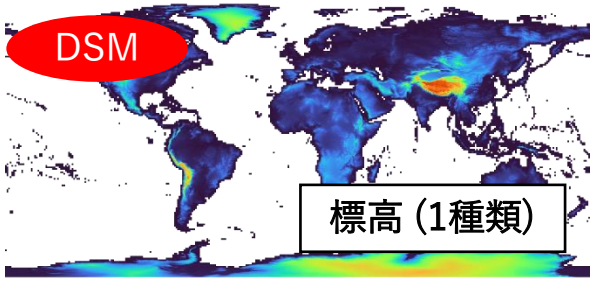
土地被覆分類図（1種類）


降水レート（5種類）

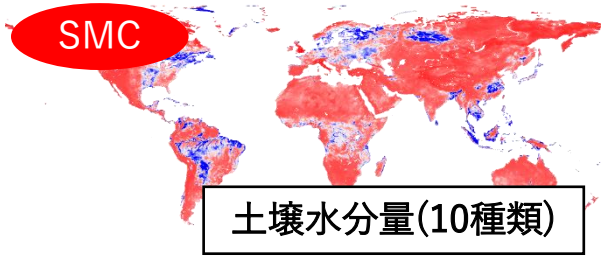


エアロゾル光学的厚さ（5種類）

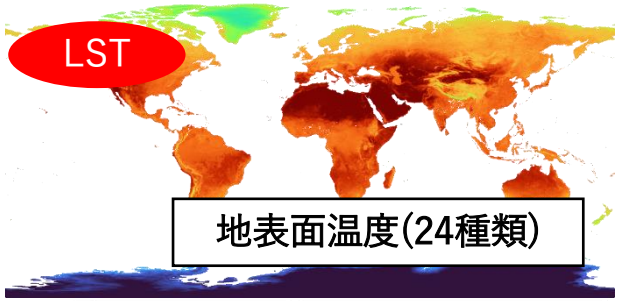
海氷密接度（1種類）



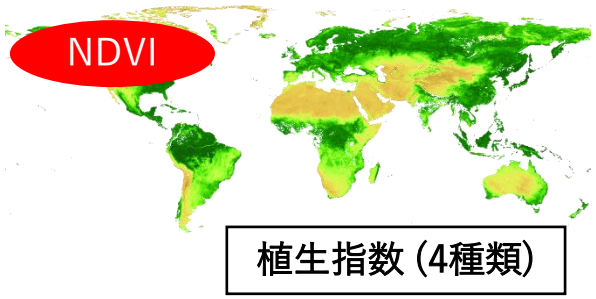
標高（1種類）



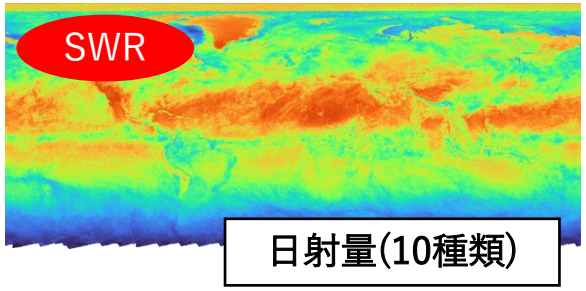
土壌水分量（10種類）



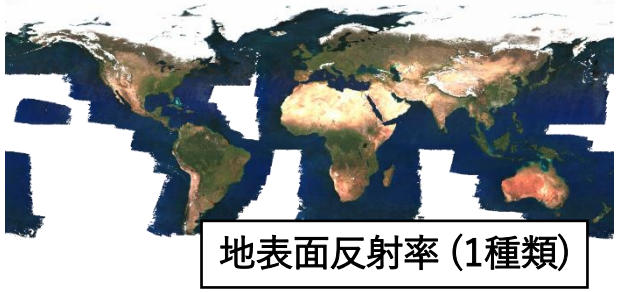
地表面温度（24種類）



植生指数（4種類）



日射量（10種類）

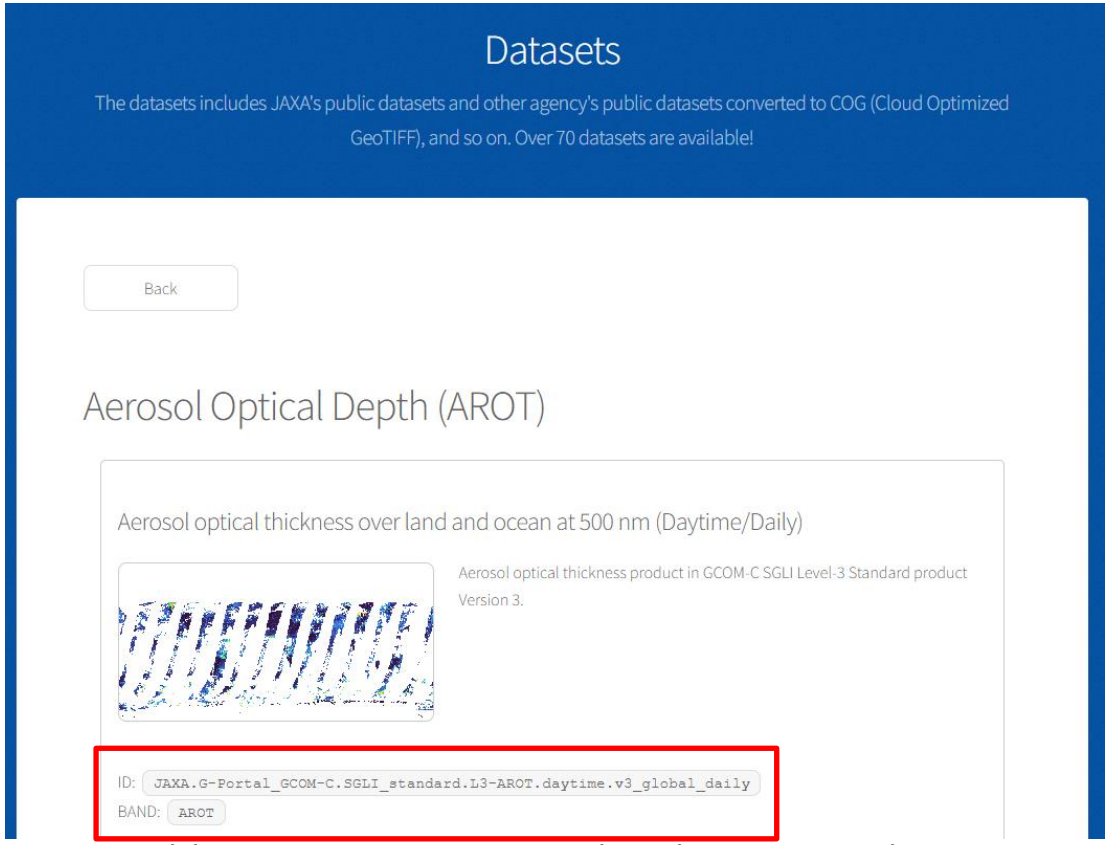

地表面反射率（1種類）

上記に示すような様々な地球環境データを取得可能（合計70種類以上）。現在はプロトタイプ版のため、限定的なデータ公開（2021年の1年間分のみ）。標高データ（AW3D）については全球、最高解像度（30m）までデータが利用可能。

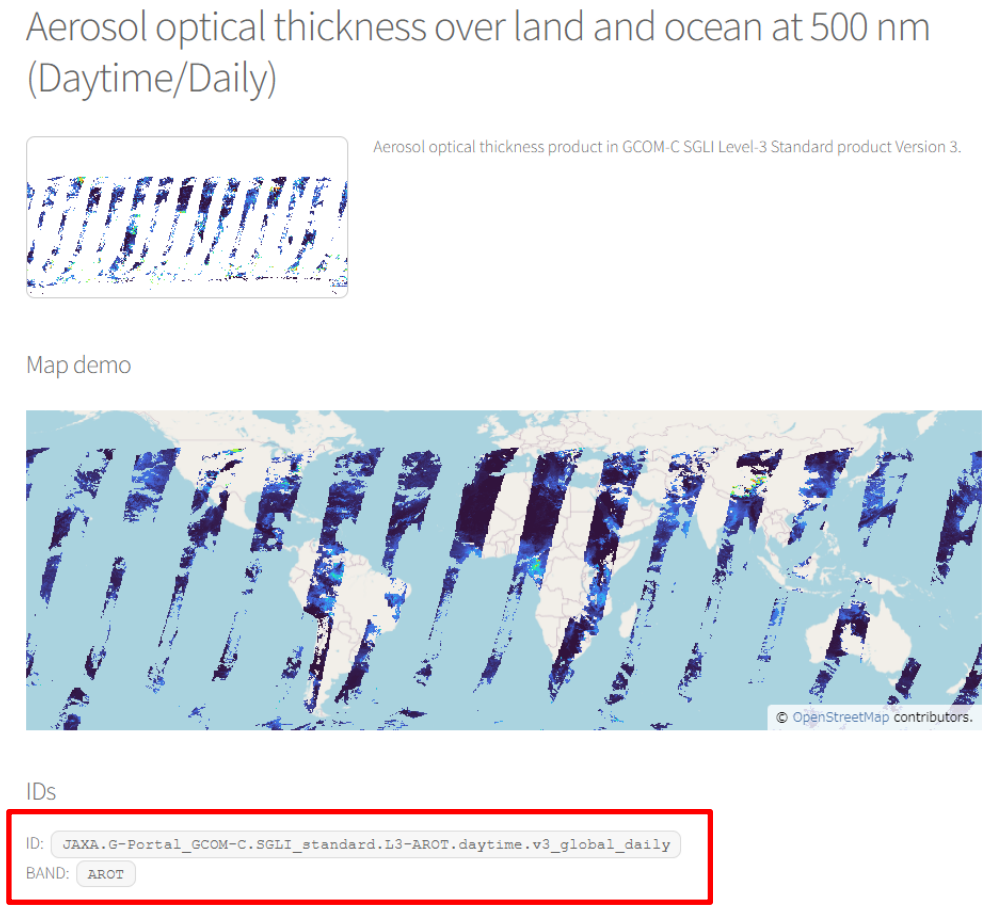
JAXA Earth API for Python：概要
- 代表的な衛星データの略称と詳細 -

正式名称(日)	正式名称(英)	略称(英)	詳細
土地被覆 分類図	Land Cover Classification System	LCCS	土地を、耕作地、都市域、その他植生等の複数のカテゴリに分類したもの
数値表層 （標高）モデル	Digital Surface Model	DSM	地形及び地形上の構造物や植生等を含む標高をデジタルデータ化したもの
降水レート	Precipitation Rate	-	単位時間あたりの降水量を示すデータ
正規化 植生指数	Normalized Difference Vegetation Index	NDVI	植物の量や活力を表すデータ。
地表面温度	Land Surface Temperature	LST	地表面の温度を示すデータ。
短波長日射量	Short Wave Radiation	SWR	可視、近紫外、近赤外の波長をもつ放射エネルギー。
土壌水分	Soil Moisture Contents	SMC	土壌に含まれる水分割合を示すデータ。
エアロゾル 光学的厚さ	Aerosol Optical Tichness	AOT	光の通りにくさを指標化したもの。

JAXA Earth API for Python：概要
- 利用可能なデータのコレクションID, Band -



https://data.earth.jaxa.jp/en/datasets/



データは、データセットに記載のコレクションIDで識別される。APIは、このID名（または含まれる文字列のキーワード）を基に、プロダクトを選定する

JAXA Earth API for Python：事前準備
- QGISの入手とインストール -

QGISとは

・地理情報システム(GIS)の閲覧、編集、分析機能を有する、無料のデスクトップアプリケーション

・QGISにはPythonが組み込まれている。

入手、インストール

・以下のサイトよりQGISを入手、インストール

https://qgis.org/ja/site/forusers/download.html

・新規ユーザーの場合は、スタンドアロンインストーラー(MSI)推奨

地図表示及びPythonコンソール表示、スクリプト実行手順

・QGIS起動後、左側の「ブラウザ」内「XYZ Tiles」の「OpenStreetMap」をダブルクリック

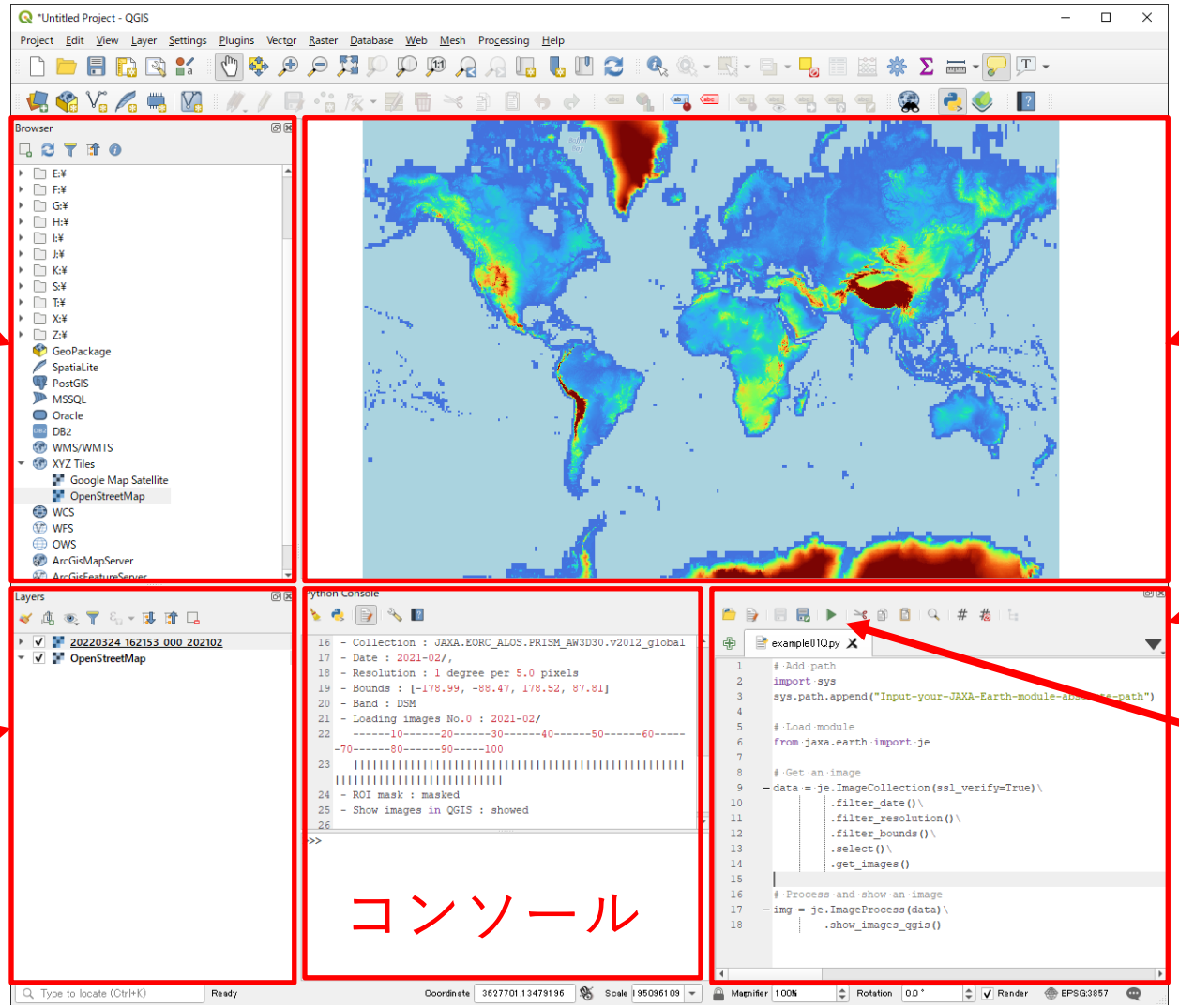
・上部メニュー「プラグイン」の「Pythonコンソール」を選択

・コンソール内「エディタの表示」  でスクリプト表示画面へ

・「ファイルを開く」または「新規作成」でPythonスクリプトを記入、修正する

・スクリプト記入後は、実行ボタン  でスクリプトを実行、衛星データ表示

JAXA Earth API for Python：事前準備
- QGISの使い方 -



データソース確認

地図表示

Python スクリプト

実行ボタン

レイヤの表示設定

コンソール

JAXA Earth API for Python：事前準備
- APIの入手、解凍、パス取得 -

APIの入手と解凍

・APIリファレンス記載の以下ページよりダウンロード

https://data.earth.jaxa.jp/api/python/quick.html

・入手後、zip解凍を行って任意の場所に保存し、そのフォルダパスを取得

（例：C:¥Users¥UserName¥Desktop¥jaxa-earth-0.1.0）

スクリプトの記載、パスの修正

・以下ページのスクリプトをQGISのスクリプトエリアにコピー＆ペースト

https://data.earth.jaxa.jp/api/python/examples2.html#get-and-show-an-image-in-qgis

・なお、スクリプトの記述内容（一部）について、APIのフォルダパスに書き換える

・**フォルダパス文字列に「¥」「＼」があるとエラーになるので、すべて「/」に書き換える**

（例： C:/Users/UserName/Desktop/jaxa-earth-0.1.0）

・Pythonスクリプトを実行する

・地図表示エリアに衛星画像が表示されれば成功

JAXA Earth API for Python：基本的な使用方法
- メソッドチェーンによるプログラミング -

```
# Method Chain sample
Taro = Human.get_up(time = “08:00”)¥
                .go_school(method = “Train”)¥
                .take_lunch(menu = “Standard A”)¥
                .go_home(method = “Train”)¥
                .sleep(time = “22:00”)
```

Humanクラス
（メソッドの塊）

get_upメソッド実行
(時刻は8時指定)

go_schoolメソッド実行
(手段は電車指定)

take_lunchメソッド実行
(メニューはA定食)

・・・

メソッドチェーン(連続的なメソッド(関数)の処理)でAPIを実行する。メソッド実行後には、その状態が保持される。
そのため、メソッドの順番は重要となる(例：学校に行く前に家に帰ることはできない)。
文字列が長くなるので、バックスラッシュ（¥）で改行する。

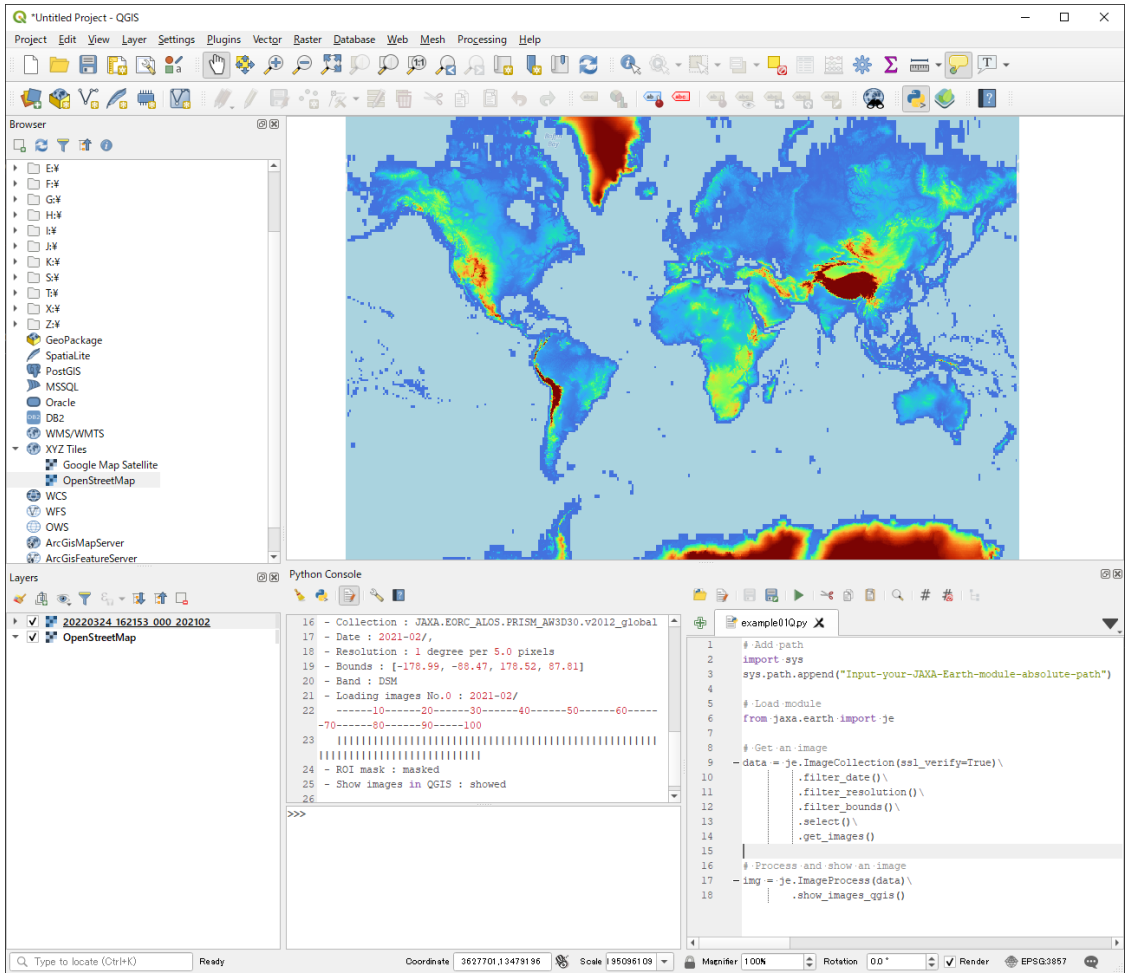
JAXA Earth API for Python：基本的な使用方法
- 画像の取得、可視化 -

```
# Add path
import sys
sys.path.append("C://YOUR-FOLDER-PATH/jaxa-earth-0.1.0")

# Load module
from jaxa.earth import je

# Get an image
data = je.ImageCollection(ssl_verify=True)¥
        .filter_date()¥
        .filter_resolution()¥
        .filter_bounds()¥
        .select()¥
        .get_images()

# Process and show an image
img = je.ImageProcess(data)¥
        .show_images_qgis()
```

注意！実際のAPIパスへ変更！
「¥」は「/」へ修正が必要

← APIの読込。毎回必要！

↑
sslエラー時はFalseへ



https://data.earth.jaxa.jp/api/python/examples2.html

QGISで標高データを可視化

JAXA Earth API for Python：基本的な使用方法
- 各種条件(プロダクト、日付範囲、領域、解像度)設定 -

```
# Add path
import sys
sys.path.append("C://YOUR-FOLDER-PATH/jaxa-earth-0.1.0")

# Load module
from jaxa.earth import je

# Set query parameters
kw   = ["Aqua","LST","half-monthly"]
dlim = ["2021-01-01T00:00:00","2021-01-01T00:00:00"]
ppu  = 5
bbox = [110, 20, 160, 50]

# Get information of collections,bands
collections,bands = je.ImageCollectionList(ssl_verify=True)¥
                    .filter_name(keywords=kw)

# Get an image
data = je.ImageCollection(collection=collections[0],ssl_verify=True)¥
       .filter_date(dlim=dlim)¥
       .filter_resolution(ppu=ppu)¥
       .filter_bounds(bbox=bbox)¥
       .select(band=bands[0][0])¥
       .get_images()

# Process and show an image
img = je.ImageProcess(data)¥
      .show_images_qgis()
```

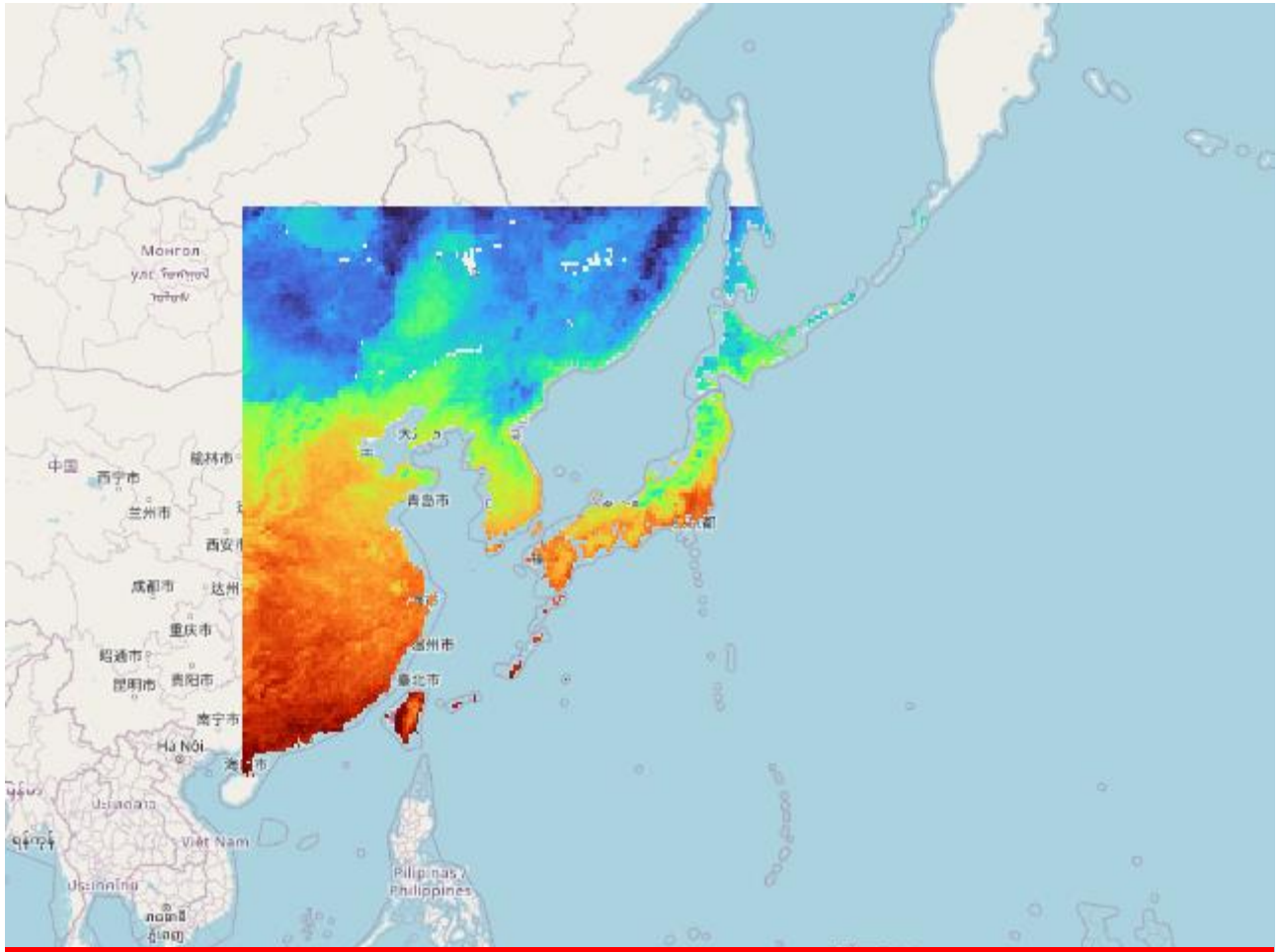
注意！実際のAPIパスへ変更！
「¥」は「/」へ修正が必要

← プロダクト特定のためのキーワード

← 日付範囲

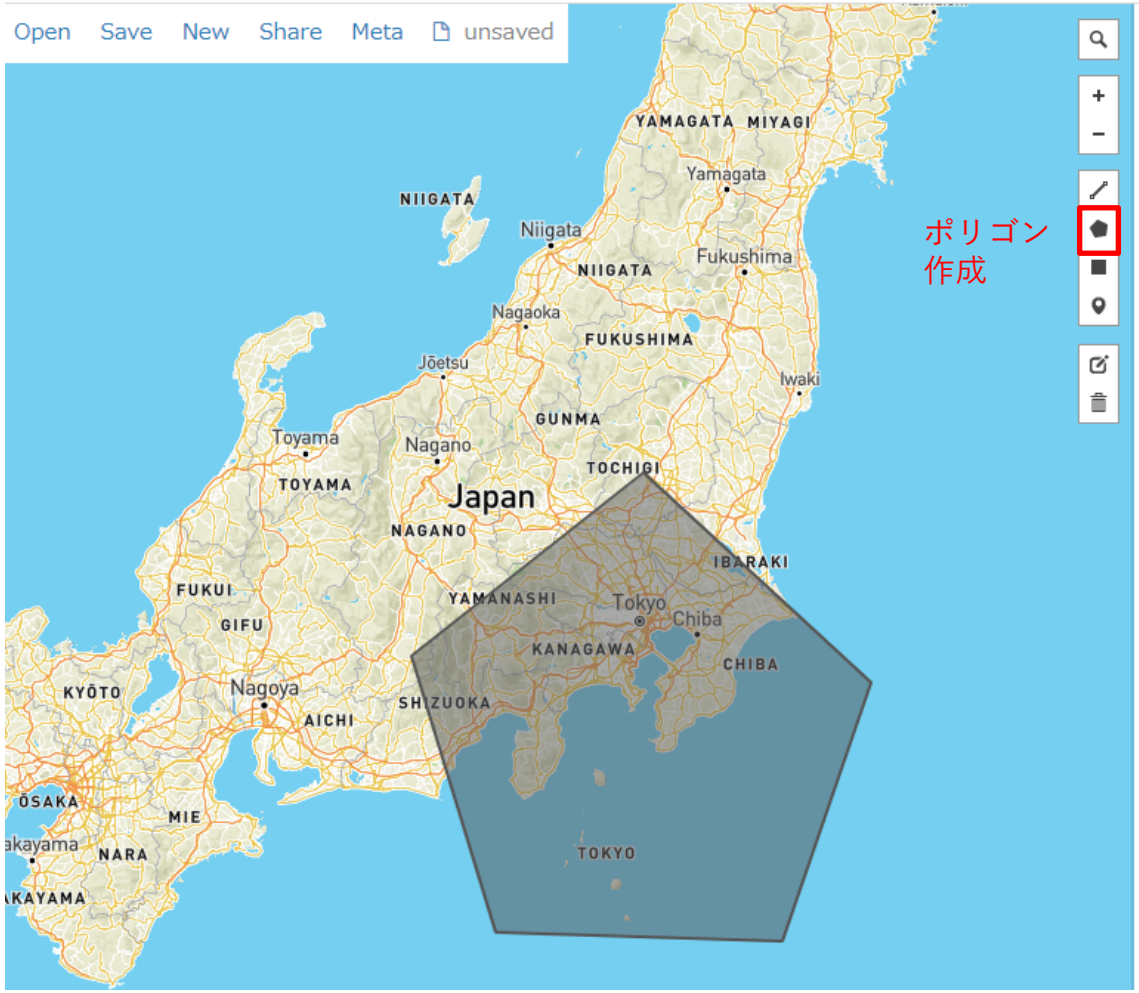
← 解像度 (1 pixel per unit (degree))

← 最小経度、最小緯度、最大経度、最大緯度



QGISで地表面温度データが可視化

JAXA Earth API for Python：基本的な使用方法
- geojsonデータの作り方 -



「https://geojson.io」にて、ポリゴン作成→Save→geojsonで、任意のエリアポリゴンを作成可能

JAXA Earth API for Python：基本的な使用方法
- geojsonによる領域指定 -

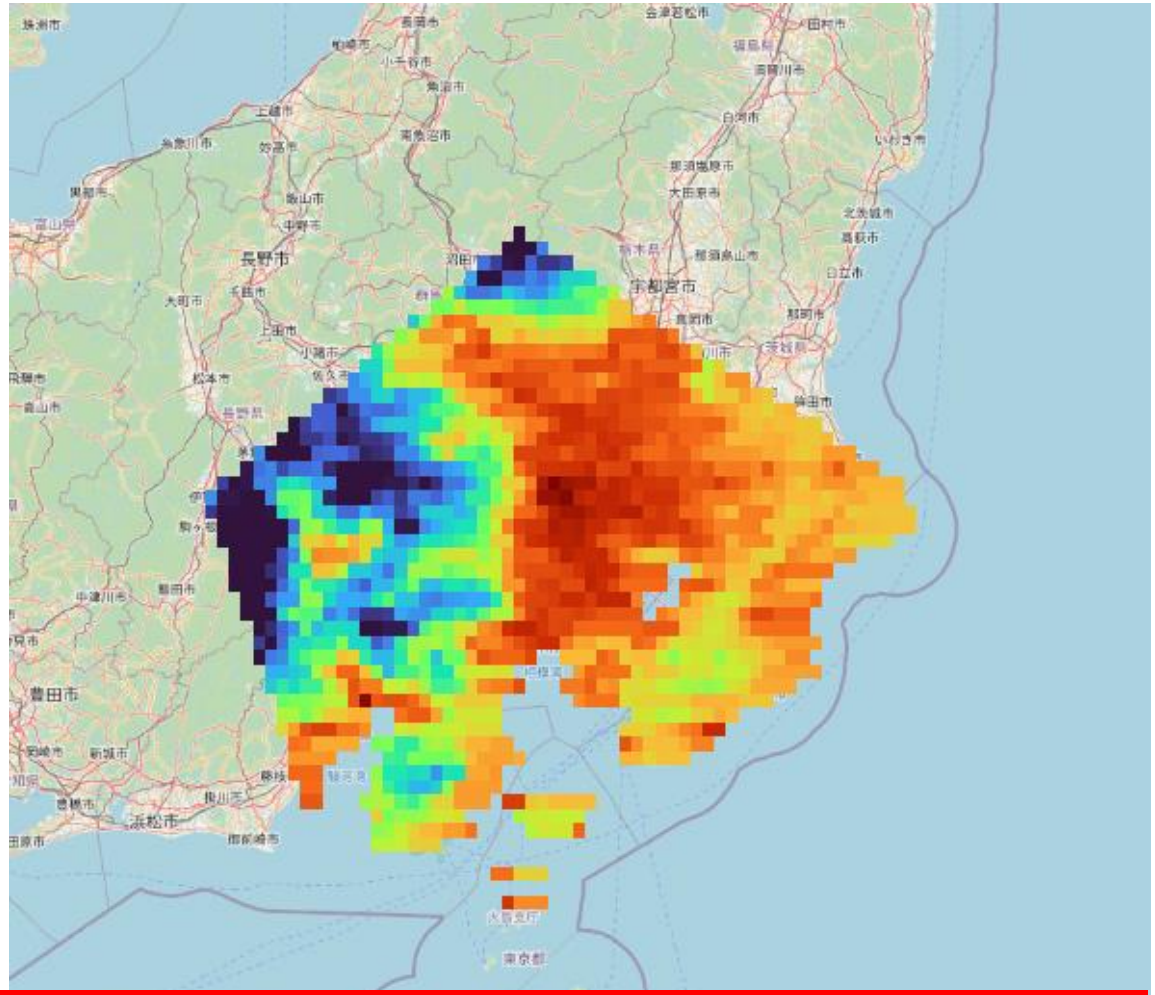
```
# Add path
import sys
sys.path.append("C://YOUR-FOLDER-PATH/jaxa-earth-0.1.0")

# Load module
from jaxa.earth import je

# Set query parameters
kw   = ["Aqua","LST","half-monthly"]
dlim = ["2021-01-01T00:00:00","2021-01-01T00:00:00"]
ppu  = 20

# Get information of collections,bands
collections,bands = je.ImageCollectionList(ssl_verify=True)¥
                    .filter_name(keywords=kw)

# Get feature collection data
geoj_path = "C://YOUR-FOLDER-PATH/jaxa-earth-0.1.0/test.geojson"
geoj = je.FeatureCollection().read(geoj_path).select()

# Get an image
data = je.ImageCollection(collection=collections[0],ssl_verify=True)¥
       .filter_date(dlim=dlim)¥
       .filter_resolution(ppu=ppu)¥
       .filter_bounds(geoj=geoj[0])¥
       .select(band=bands[0][0])¥
       .get_images()

# Process and show an image
img = je.ImageProcess(data)¥
      .show_images_qgis()
```

注意！実際のAPIパスへ変更！
「¥」は「/」へ修正が必要

注意！実際のAPIパスへ変更！
「¥」は「/」へ修正が必要

← test.geojsonを読込

← geojsonによる領域指定



作成したポリゴン内のみのデータ(地表面温度)が可視化

JAXA Earth API for Python：基本的な使用方法
- 参考：土地被覆分類図のラベルと値の関係 -

Label		Value		Color	RGB
Global	Regional	Global	Regional		
No Data		0			0, 0, 0
Cropland, rainfed		10			255, 255, 100
耕作地	Cropland, rainfed, herbaceous cover		11		255, 255, 100
	Cropland, rainfed, tree or shrub cover		12		255, 255, 0
Cropland, irrigated or post-flooding		20			170, 240, 240
Mosaic cropland (>50%) / natural vegetation (tree, shrub, herbaceous cover) (<50%)		30			220, 240, 100
Mosaic natural vegetation (tree, shrub, herbaceous cover) (>50%) / cropland (<50%)		40			200, 200, 100
Tree cover, broadleaved, evergreen, closed to open (>15%)		50			0, 100, 0
Tree cover, broadleaved, deciduous, closed to open (>15%)		60			0, 160, 0
	Tree cover, broadleaved, deciduous, closed (>40%)		61		0, 160, 0
	Tree cover, broadleaved, deciduous, open (15-40%)		62		170, 200, 0
Tree cover, needleleaved, evergreen, closed to open (>15%)		70			0, 60, 0
	Tree cover, needleleaved, evergreen, closed (>40%)		71		0, 60, 0
	Tree cover, needleleaved, evergreen, open (15-40%)		72		0, 80, 0
Tree cover, needleleaved, deciduous, closed to open (>15%)		80			40, 80, 0
	Tree cover, needleleaved, deciduous, closed (>40%)		81		40, 80, 0
	Tree cover, needleleaved, deciduous, open (15-40%)		82		40, 100, 0
Tree cover, mixed leaf type (broadleaved and needleleaved)		90			120, 130, 0
Mosaic tree and shrub (>50%) / herbaceous cover (<50%)		100			140, 160, 0
Mosaic herbaceous cover (>50%) / tree and shrub (<50%)		110			190, 150, 0
Shrubland		120			150, 100, 0
	Evergreen shrubland		121		150, 100, 0
	Deciduous shrubland		122		150, 100, 0
Grassland		130			255, 180, 50
Lichens and mosses		140			255, 220, 210
Sparse vegetation (tree, shrub, herbaceous cover) (<15%)		150			255, 235, 175
	Sparse tree (<15%)		151		255, 200, 100
	Sparse shrub (<15%)		152		255, 210, 120
	Sparse herbaceous cover (<15%)		153		255, 235, 175
Tree cover, flooded, fresh or brackish water		160			0, 120, 90
Tree cover, flooded, saline water		170			0, 150, 120
Shrub or herbaceous cover, flooded, fresh/saline/brackish water		180			0, 220, 130
Urban areas 都市域		190			195, 20, 0
Bare areas		200			255, 245, 215
	Consolidated bare areas	201			220, 220, 220
	Unconsolidated bare areas	202			255, 245, 215
Water bodies		210			0, 70, 200
Permanent snow and ice		220			255, 255, 255

Product User Guide and Specification ICDR Land Cover 2016 to 2019

JAXA Earth API for Python：基本的な使用方法
- 土地被覆分類による領域マスク -

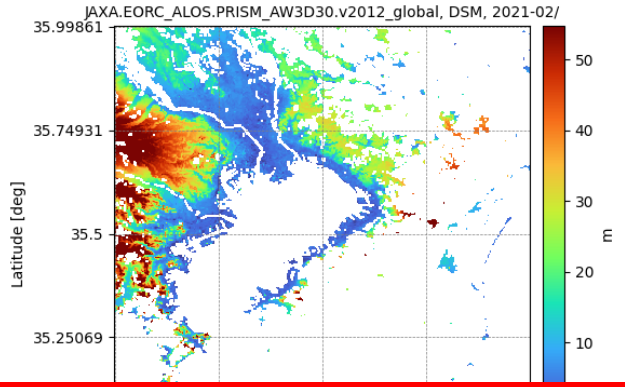
```
# Add path
import sys
sys.path.append("C://YOUR-FOLDER-PATH/jaxa-earth-0.1.0")

# Load module
from jaxa.earth import je

# Set query parameters
kw_d = ["AW3D"]
kw_m = ["LCCS"]
dlim = ["2019-01-01T00:00:00","2021-02-01T00:00:00"]
ppu  = 360
bbox = [139.5, 35, 140.5, 36]
mq   = "values_equal"
val  = [190]

# Get information of collections,bands for data
collections_d,bands_d = je.ImageCollectionList(ssl_verify=True)¥
                .filter_name(keywords=kw_d)

# Get an image for data
data_d = je.ImageCollection(collection=collections_d[0],ssl_verify=True)¥
        .filter_date(dlim=dlim)¥
        .filter_resolution(ppu=ppu)¥
        .filter_bounds(bbox=bbox)¥
        .select(band=bands_d[0][0])¥
        .get_images()
```

注意！実際のAPIパスへ変更！
「¥」は「/」へ修正が必要

← 標高
← 土地被覆分類図
← 「値が等しい」ピクセルを抽出
← 土地被覆分類図の値が190 = 都市域
← 標高データを取得

```
# Get information of collections,bands for mask
collections_m,bands_m = je.ImageCollectionList(ssl_verify=True)¥
                .filter_name(keywords=kw_m)

# Get an image for mask
data_m = je.ImageCollection(collection=collections_m[0],ssl_verify=True)¥
        .filter_date(dlim=dlim)¥
        .filter_resolution(ppu=ppu)¥
        .filter_bounds(bbox=bbox)¥
        .select(band=bands_m[0][0])¥
        .get_images()

# Process and show an image
img = je.ImageProcess(data_d)¥
      .mask_images(data_m,method_query=mq, values=val)¥
      .show_images_qgis()
```

← 土地被覆データを取得
← マスク処理、可視化



指定した土地被覆分類(都市域)のみの標高データを可視化

JAXA Earth API for Python：基本的な使用方法
- 時系列データ処理、取得 -

```
# Add path
import sys
sys.path.append("C://YOUR-FOLDER-PATH/jaxa-earth-0.1.0")

# Load module
from jaxa.earth import je

# Set query parameters
kw_d = ["swr","half-monthly-normal"]
dlim = ["2021-01-01T00:00:00","2021-12-31T00:00:00"]
bbox = [120,20,150,50]
ppu  = 10

# Get information of collections,bands for data
collections,bands = je.ImageCollectionList(ssl_verify=True)¥
                    .filter_name(keywords=kw_d)

# Get an image for data
data = je.ImageCollection(collection=collections[0],ssl_verify=True)¥
        .filter_date(dlim=dlim)¥
        .filter_resolution(ppu=ppu)¥
        .filter_bounds(bbox=bbox)¥
        .select(band=bands[0][0])¥
        .get_images()

# Process and show an image
img = je.ImageProcess(data)¥
      .calc_spatial_stats()¥
      .show_spatial_stats()
```

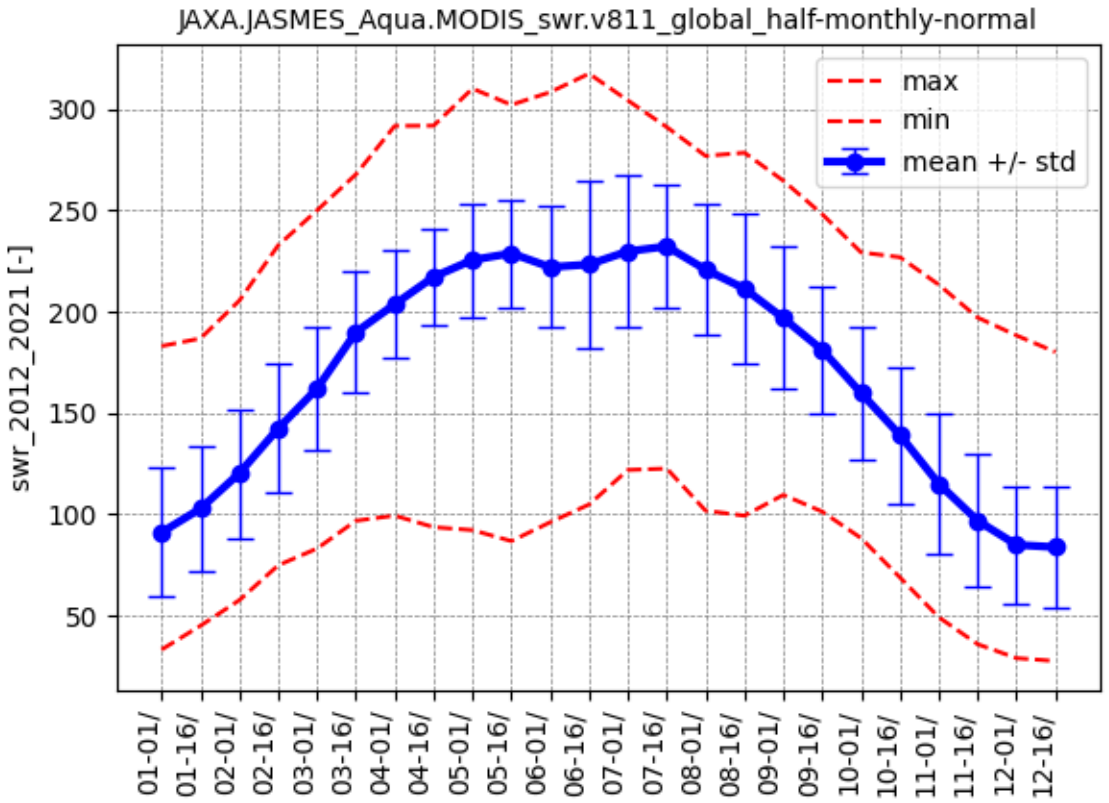
注意！実際のAPIパスへ変更！
「¥」は「/」へ修正が必要

← 期間を1年に設定

← 複数枚の衛星画像を取得

← 画像を空間統計処理

← 時系列グラフ表示



日本付近の日射量データを時系列グラフ化

JAXA Earth API for Python ：データ利用と簡単な分析事例
- 国、地域、季節の違いによる地球環境差異の把握

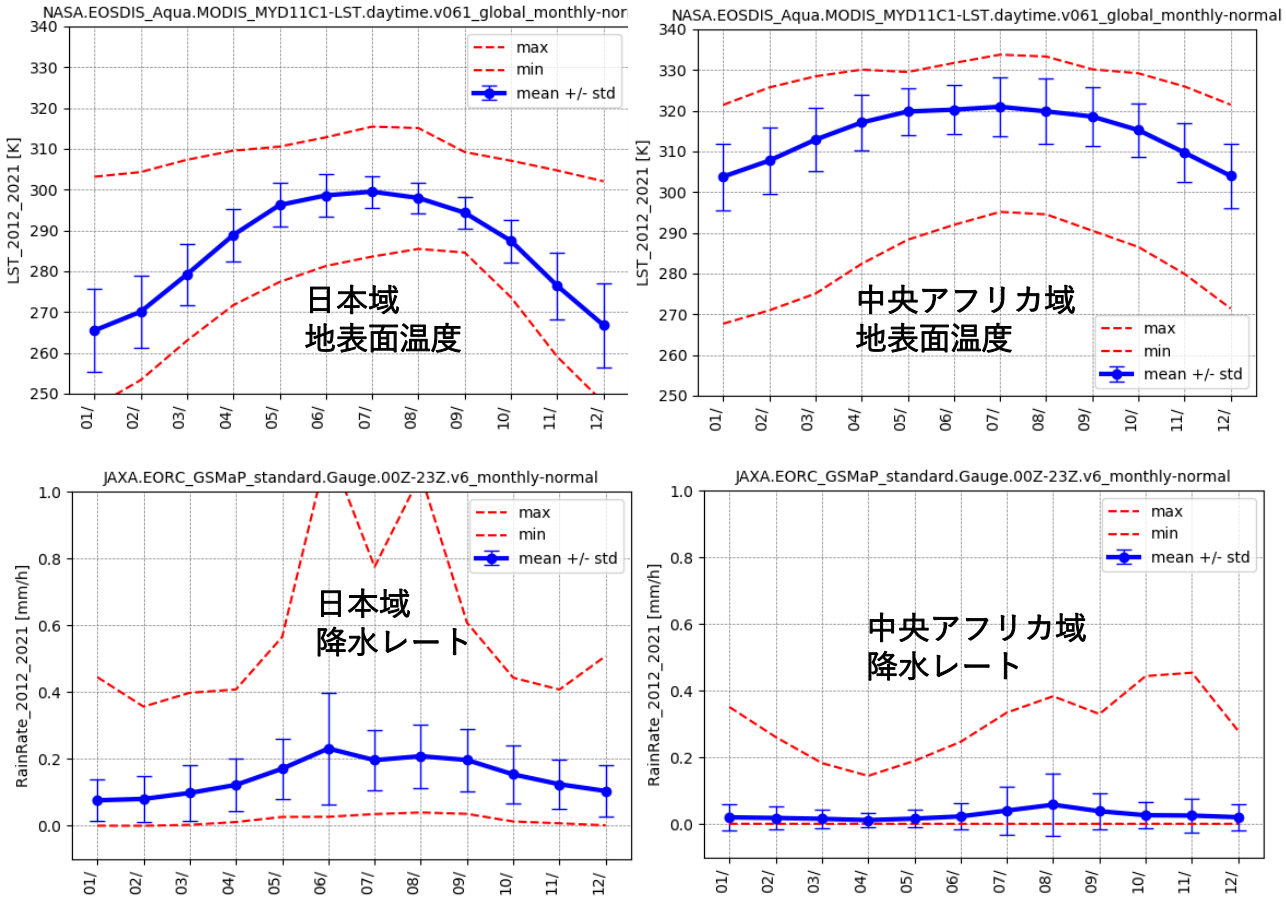
```
# Add path
import sys
sys.path.append("C://YOUR-FOLDER-PATH/jaxa-earth-0.1.0")

# Load module
from jaxa.earth import je

# Set query parameters
#kw_d = ["LST","_monthly-normal"]
kw_d = ["GSM","_monthly-normal"]
dlim = ["2021-01-01T00:00:00","2021-12-31T00:00:00"]
#bbox = [120,20,150,50] # Japan
bbox = [0,10,30,40] # Africa
ylim  = [-0.1,1] # Rain
ylim = [250,340] # LST
ppu  = 10

# Get information of collections,bands for data
collections,bands = je.ImageCollectionList(ssl_verify=True)¥
                    .filter_name(keywords=kw_d)

# Get an image for data
data = je.ImageCollection(collection=collections[0],ssl_verify=True)¥
        .filter_date(dlim=dlim)¥
        .filter_resolution(ppu=ppu)¥
        .filter_bounds(bbox=bbox)¥
        .select(band=bands[0][0])¥
        .get_images()

# Process and show an image
img = je.ImageProcess(data)¥
      .calc_spatial_stats()¥
      .show_spatial_stats()
```

注意！実際のAPIパスへ変更！
「¥」は「/」へ修正が必要

← 複数枚の衛星画像を取得

← 画像を空間統計処理

← 時系列グラフ表示



各地域(日本、アフリカ)の平均的な地表面温度、降水レートを可視化

JAXA Earth API for Python ：データ利用と簡単な分析事例 - 農業、生態分布把握等への利用

```
# Add path
import sys
sys.path.append("C://YOUR-FOLDER-PATH/jaxa-earth-0.1.0")

# Load module
from jaxa.earth import je

# Set query parameters
kw_d = ["LST","_monthly-normal"]
ylim = [250,320]
#kw_d = ["ndvi","_monthly-normal"]
#ylim = [0,1]
kw_m = ["LCCS"]
dlim_d = ["2021-01-01T00:00:00","2021-12-01T00:00:00"]
dlim_m = ["2019-01-01T00:00:00","2021-12-01T00:00:00"]
ppu  = 20
bbox = [120, 20, 150, 50]
mq   = "values_equal"
#val  = [190]
val  = [10,11,12]

# Get information of collections,bands for data
collections_d,bands_d = je.ImageCollectionList(ssl_verify=True)¥
                .filter_name(keywords=kw_d)
```

注意！実際のAPIパスへ変更！
「¥」は「/」へ修正が必要

← 地表面温度用キーワード

← 植生指数用キーワード

← 土地被覆分類図用キーワード

← 190 : 都市域

← 10,11,12 : 耕作域

```
# Get an image for data
data_d = je.ImageCollection(collection=collections_d[0],ssl_verify=True)¥
        .filter_date(dlim=dlim_d)¥
        .filter_resolution(ppu=ppu)¥
        .filter_bounds(bbox=bbox)¥
        .select(band=bands_d[0][0])¥
        .get_images()

# Get information of collections,bands for mask
collections_m,bands_m = je.ImageCollectionList(ssl_verify=True)¥
                .filter_name(keywords=kw_m)

# Get an image for mask
data_m = je.ImageCollection(collection=collections_m[0],ssl_verify=True)¥
        .filter_date(dlim=dlim_m)¥
        .filter_resolution(ppu=ppu)¥
        .filter_bounds(bbox=bbox)¥
        .select(band=bands_m[0][0])¥
        .get_images()

# Process and show an image
img = je.ImageProcess(data_d)¥
      .mask_images(data_m,method_query=mq, values=val)¥
      .calc_spatial_stats()¥
      .show_spatial_stats(ylim = ylim)
```

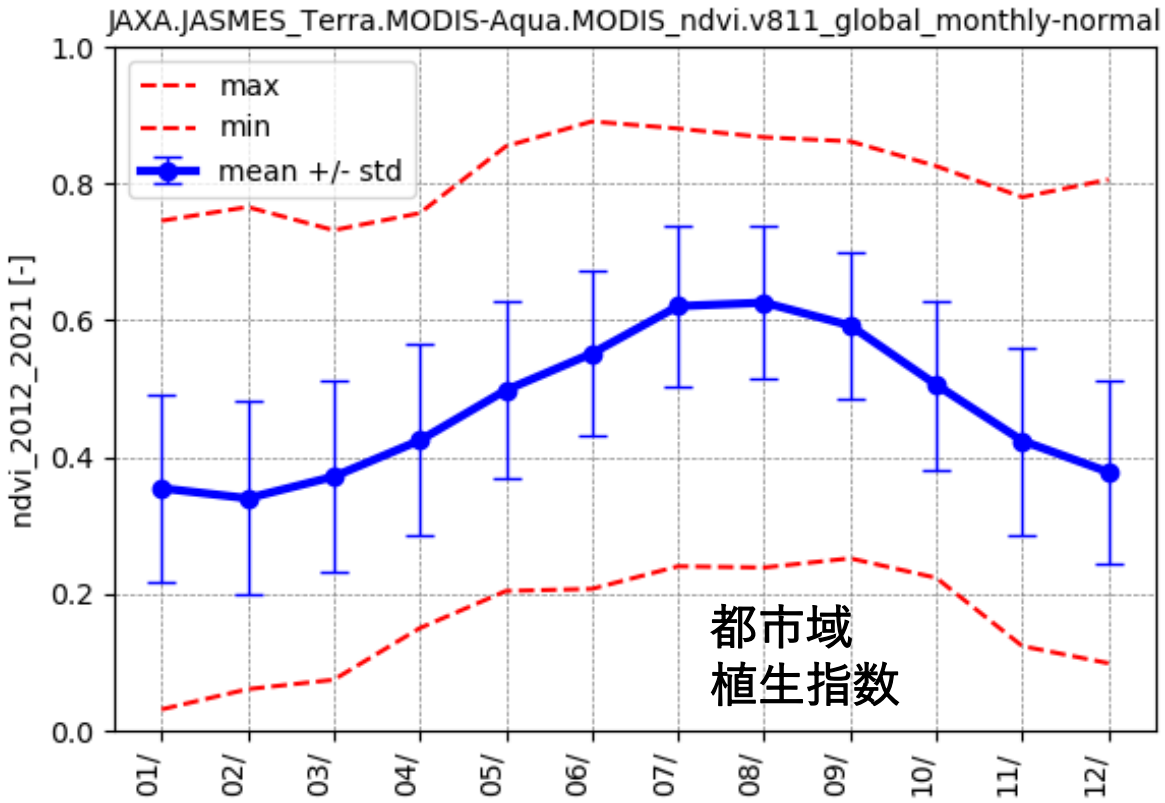
← マスク処理

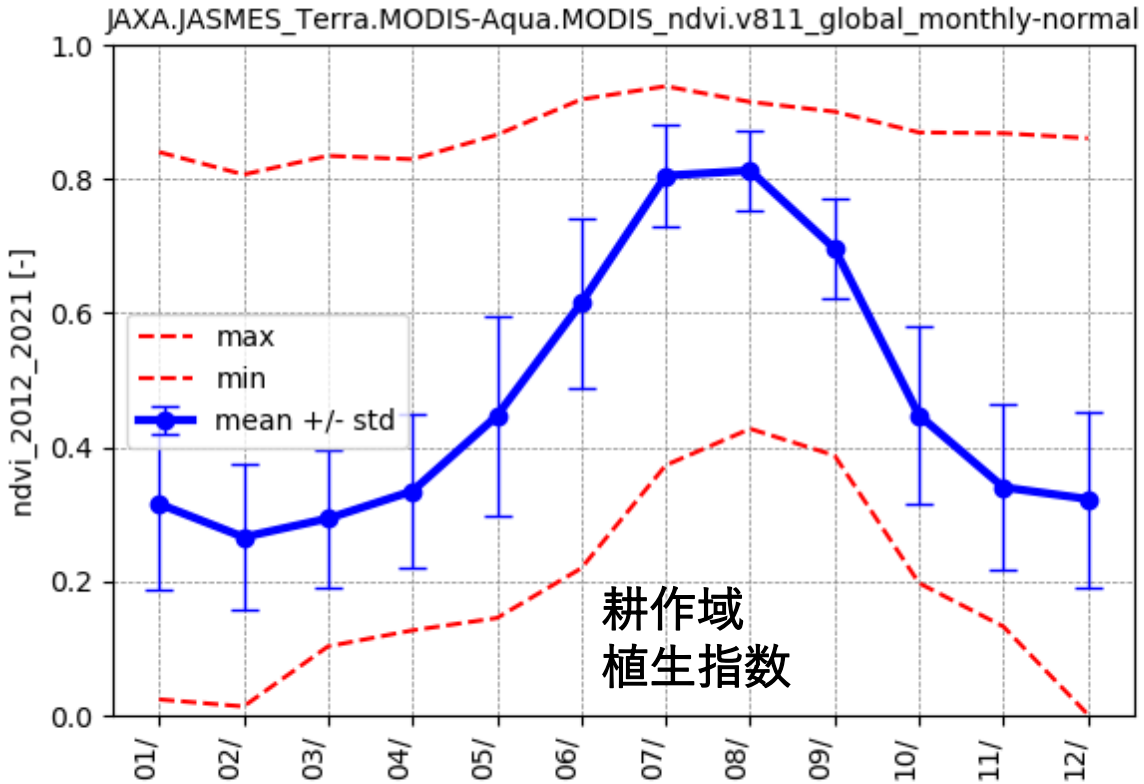
← 画像を空間統計処理

← 時系列グラフ表示

都市域と耕作地の植生指数、地表面温度の推移を確認

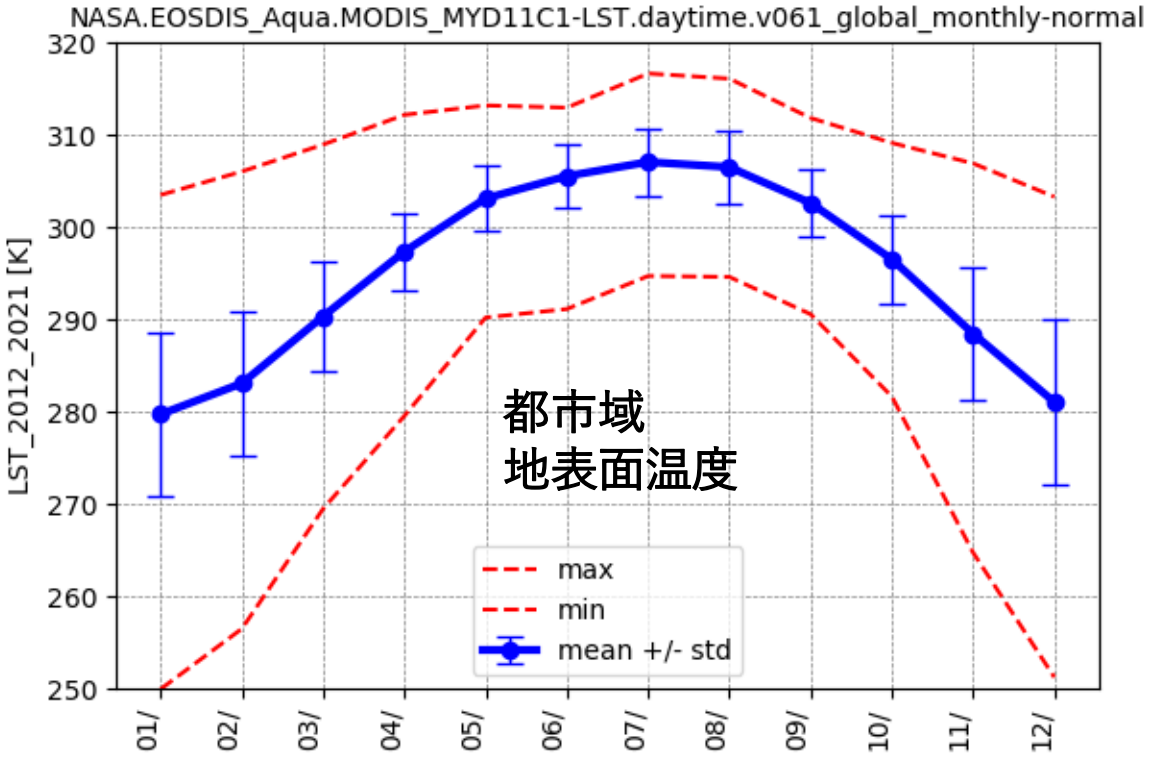
JAXA Earth API for Python ：データ利用と簡単な分析事例
- 農業、生態分布把握等への利用

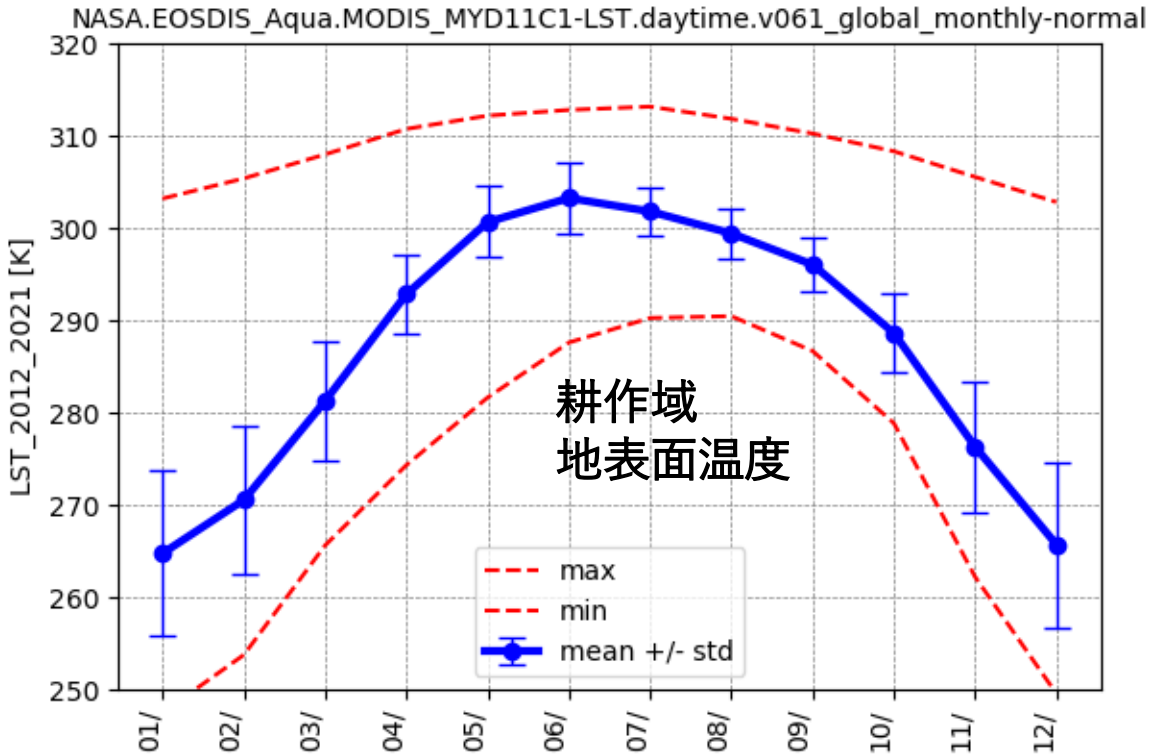




都市域と耕作地の植生指数の推移を確認（夏季は耕作域が指数が高くなる）

JAXA Earth API for Python ：データ利用と簡単な分析事例
- 農業、生態分布把握等への利用





都市域と耕作地の地表面温度の推移を確認（耕作域は、通年、都市域に比べて地表面温度が低い傾向にある）

まとめ、今後の予定

まとめ

・JAXAの様々な地球観測衛星画像データを取得、処理が可能なPython APIを開発、公開している。

・Python APIを用いれば、衛星画像自動取得、マスク処理、時空間統計処理機能等の処理が容易に実行可能。

・APIのQGISへのインターフェース機能を生かすことで、よりインタラクティブなデータ閲覧、解析が可能。

今後の予定

・さらなるデータベースの拡充やアップデート、Python APIの高度化、Webブラウザアプリの開発、公開を予定。