人工衛星から見える地球環境変化を調べてみよう
～Google Earth Engine 篇～

山本雄平・渡辺玲奈・小菅生文音・市井和仁
（千葉大学環境リモートセンシング研究センター）

東京都教育委員会主催
令和 7年度　得意な才能を伸ばす教育（理数）　課題研究プログラム

目次

第１章

Google Earth Engine の簡単な紹介

1.1. はじめに

地球観測衛星は、降水、大気分子、陸面･海面温度、地殻変動など様々な状態を、様々な観測幾何条件から、様々なセンサ(光学センサ･マイクロ波センサ等)を使って観測する。それらの情報は、地球環境研究だけでなく、防災、交通・物流、農林水産業など多くの場面で活用され、今や学術研究と社会において必要不可欠なものとなっている。今後も衛星観測技術の発達やデータ蓄積量の増加、観測対象の多様化に伴って、その需要はますます高まっていくことだろう。しかしその一方で、衛星データの莫大さや処理の複雑さが“手軽な”利用促進を阻んでいるという側面もある。衛星データの利用には、大規模ストレージ環境の整備が必要なのはもちろんのこと、利用に至るまでにも、各衛星特有の座標系とデータフォーマットに応じた前処理を行わなければならない。またこれを行うためのプログラミング言語や地理情報システム（GIS）の知識は必須となる。

GoogleEarth Engine（GEE)は、高度なプログラミング技術を要することなく、またストレージやメモリの容量も気にせず衛星データを手軽に扱える画期的なプラットフォームである。初心者でも扱いやすいインタフェースが提供されており、様々な衛星データセットが整った形式で Google のクラウドストレージ上に置かれ、かつそれらの解析計算も Google サーバ上で行われる。つまり GEE に接続できるインターネット環境さえあれば、どのスペックの PC でも膨大な衛星データ処理が可能なわけである。

GEE のメインページの URL は、https://earthengine.google.com/ である。まずはこのサイトを見てみよう。図１のような画面が出てくるので、アカウントがない場合は Get Started（図１ 黄矢印）から作成しよう。Gmailアカウントがあればスムーズに進むだろう。ただし、アカウントの作成には長いと数日かかることもある。

次に、Watch Video(図１緑矢印)をクリックすることでGEEの紹介動画を視聴できる。ここでは、GEEの概要と、どのようなデータセットが格納されているか、どのような解析ができるのかがまとめられている。

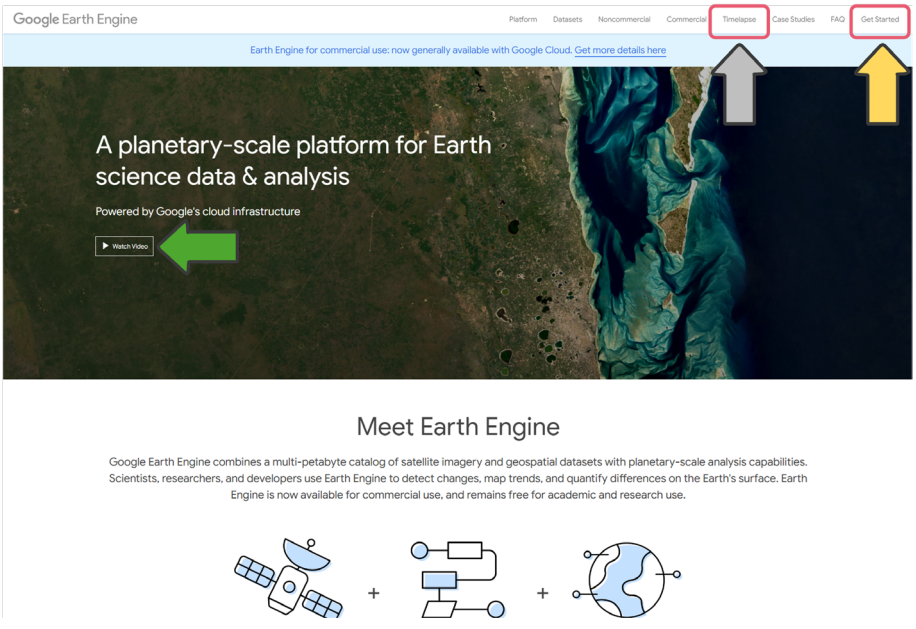

図 1. Google Earth Engine のメインページ (https://earthengine.google.com/)

1.2. Time-lapse

GEE では様々なデータセットが準備されているが、まず最も簡単に衛星データを閲覧できる “Timelapse” 機能を使って、衛星リモートセンシングによりどのような地表面変動が見えるか把握してみよう。図 1 の灰色矢印が指す箇所（Timelapse）をクリックすると、図 2 のようなページが初期画面として表示される。

Timelapse では、1984 年から 2022 年にかけての地表面の変化を見ることができる。実際に使用されている地球観測衛星の種類や画像の表示方法（描画色や観測波長帯の設定）などの詳細なカスタマイズはできないところが難点ではあるが、「簡単に世界中の地表面の変化を見ることができる」という点で非常に有用である。

試しに、初期画面（図 2）でいくつか試してみよう。この画面では、北米（アラスカ）における氷河の広がりについて、1984 年から 2022 年までの推移をみることができる。図 2 のようにズームイン・ズームアウトや、動画の速度などいくつかの調整が可能である。

また、左には世界中の様々な地域の興味深い変化がリストアップされており、クリックして確認することができる。この中のいくつかを自身でも確認してみて、衛星から見える地表面の変化のイメージをつかんでもらいたい。場所の選択、ズームインなどの基本的な機能については、各自で試してもらいたい。

左の場所のメニュー以外にも、興味深い地域は多くある。例えば、日本の東京湾は時間推移が大きい地域の一つとして挙げられる。東京国際空港など東京湾の埋め立ての状況、東京ディズニーシーの開業 (2001 年) に伴う変化など、様々な事象を眺めることができる。

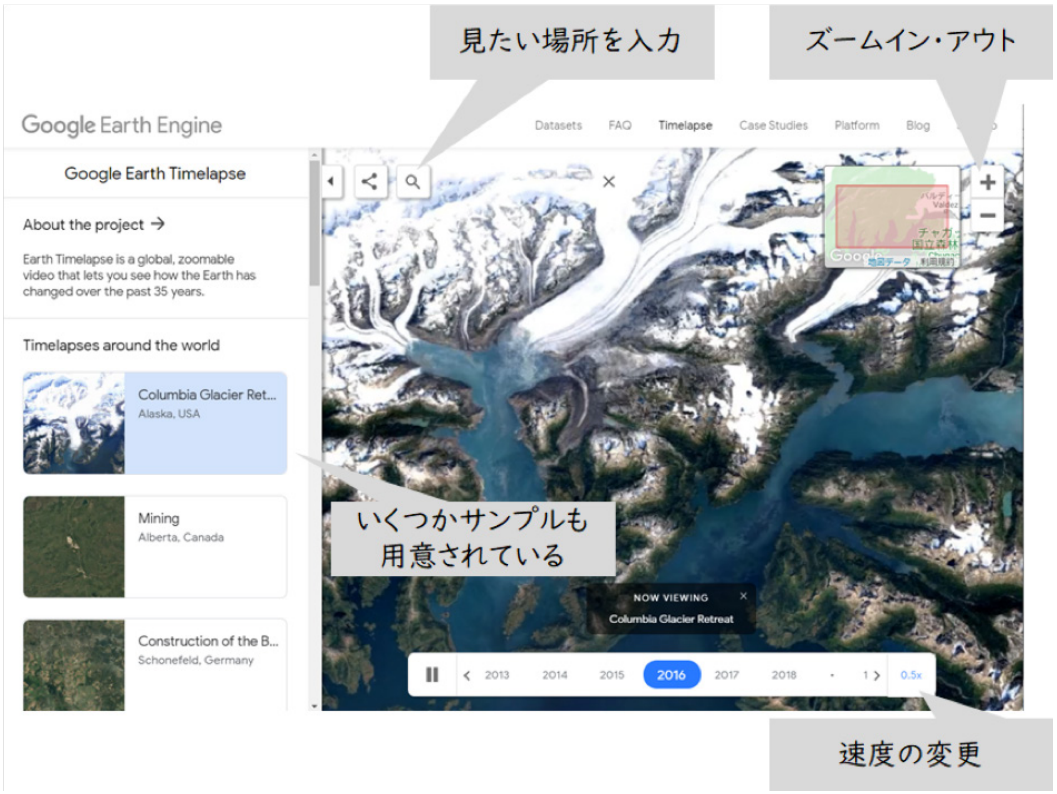

図 2. Google Earth Engine (Timelapse) の初期画面

1.3. データの検索

GEE に格納されている衛星データセットを実際に使ってみよう。図 1 の初期画面の Datasets をクリックすると、以下のようなデータカタログページが表示される（図 3）。衛星データセットはカテゴリ別に分けられており、ここから探すこともできるが、既に目当てのプロダクト名が分かっている場合は、図 3 右上の Search からキーワードを入れて検索した方が手っ取り早いだろう。

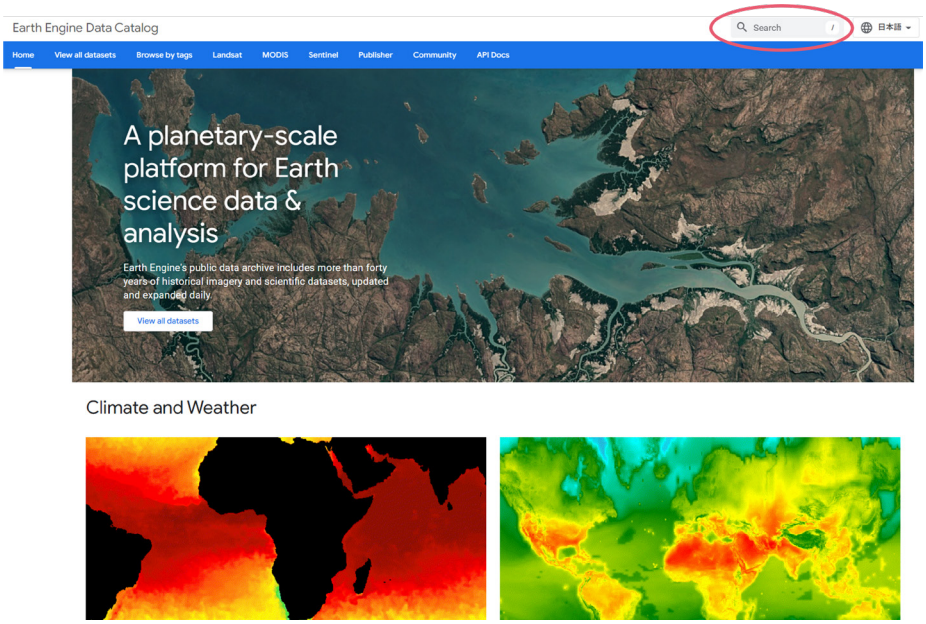

図 3. Earth Engine Data Catalog の画面

ここでは、キーワードとして「Land Surface Temperature（地表面温度）」を入れてみよう。検索結果の中で、「MOD11A2.061 Terra Land Surface Temperature and Emissivity 8-Day Global 1km 」を選択してみる。Description を見るとデータセットの説明があり、Bands タブをクリックすると、データに関する詳細な説明が表示される（図 4 ）。

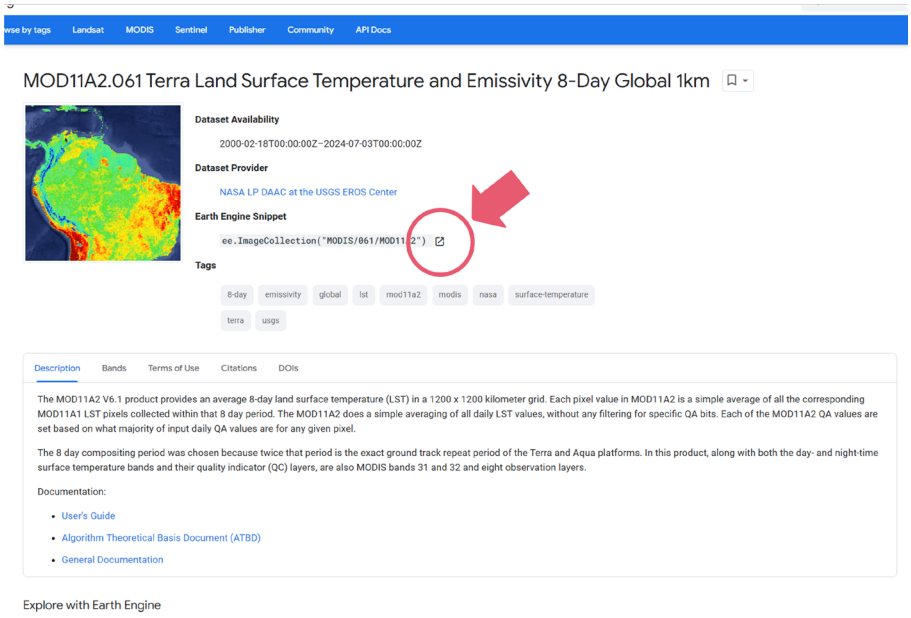

図 4. 地表面温度データに関する情報画面

1.4. インポート

検索したデータを Google Earth Engine 上で直接使うには、Earth Engine Snippet のところにある四角い印をクリックすればよい(図 4 〇部分)。すると、コード編集 & 実行の画面に移る(図 5)。真ん中上部にある文字列は、衛星データの取得･解析･描画を行うための GEE 用のプログラムである。試しに〇の Run をクリックしてみよう。下の地図上に地表面温度の分布が上塗りされるように表示されただろう（図 6)。

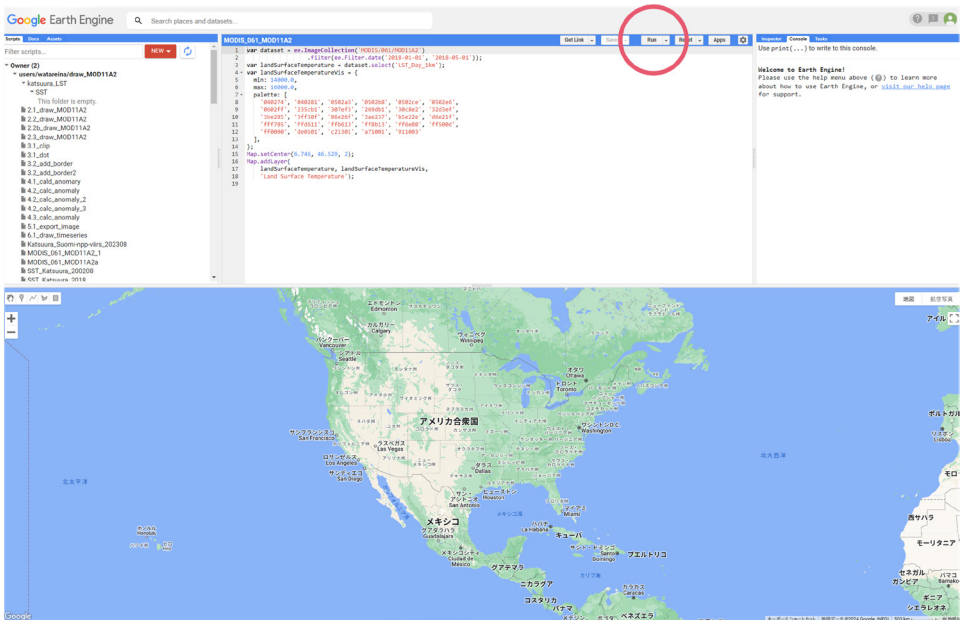

図 5. Google Earth Engine のコード編集・実行画面（Code Editor）

第 2 章以降でプログラムの扱い方を解説する前に、Code Editor の機能について触れておく。プログラムを挟んだ両端のボックスにはそれぞれ 3 つのタブ「Scripts」「Docs」「Assets」、「Inspector」「Console」「Tasks」がある。左側はスクリプトの作成・管理に関わるボックスであり、右側はスクリプトの実行結果に関わるボックスである。

Scripts　：自身で作成・編集したスクリプト（プログラムが書かれたもの）を “Owner” で管理できる。また、“Examples” では様々な解析ケースに応じたスクリプト例が用意されている。これらのスクリプト例を雛型として、本来であれば敷居の高い衛星データ解析を単純な編集だけで行えるのが GEE の長所といえる。本テキストでは扱わない複雑な解析を行う際には、コードをゼロから書き始める前に、まずはここで参考になるスクリプトがあるか調べてみよう。

Docs　：解析に用いられる様々な関数の説明が記載されている。上部の検索窓 “Filter Methods ...” で目当ての関数の使い方を調べることができる。

Assets　：ローカル PC 上で作成したスクリプトや画像などを “Owner” 内のフォルダにアップロードできる。

Inspector：下の地図上で任意の地点をクリックすると、その地点の位置情報等が表示される。Run でスクリプトを実行した後にクリックすれば、描画した衛星データの情報も表示される。図 6 はスクリプト実行後に千葉県のある地点(東経 139.97°、北緯 35.921°)をクリックした際の出力結果である。「Point」にはクリックしたピクセルの位置情報などが表示されており、「Pixels」にはデータセットのどのバンドが使われているか（ここでは Land Surface Temperature のバンド 1 「LST_Day_1km」)、また何枚の衛星画像が地図上に上塗りされているか(ここでは 15 枚)が表示されている。「Objects」には使われている衛星画像の属性（id は作成開始日にあたる）が表示されている。「Pixels」内の

「Series」をクリックしてみると、図 6 上のようにクリックした点の地表面温度の時系列グラフが表示される。解析期間の設定方法や縦軸の値（温度なのになぜ 10000 以上になる？）など、現時点ではわからない点が多くあると思うが、これらは第 2 章以降でじっくり解説する。

Console　：プログラムで指定した文字列やグラフの出力結果等が表示される（第 4 章と第 6 章を参照）。

Tasks　　：画像を出力する際に出力先や解像度等を設定する（第 5 章を参照）。

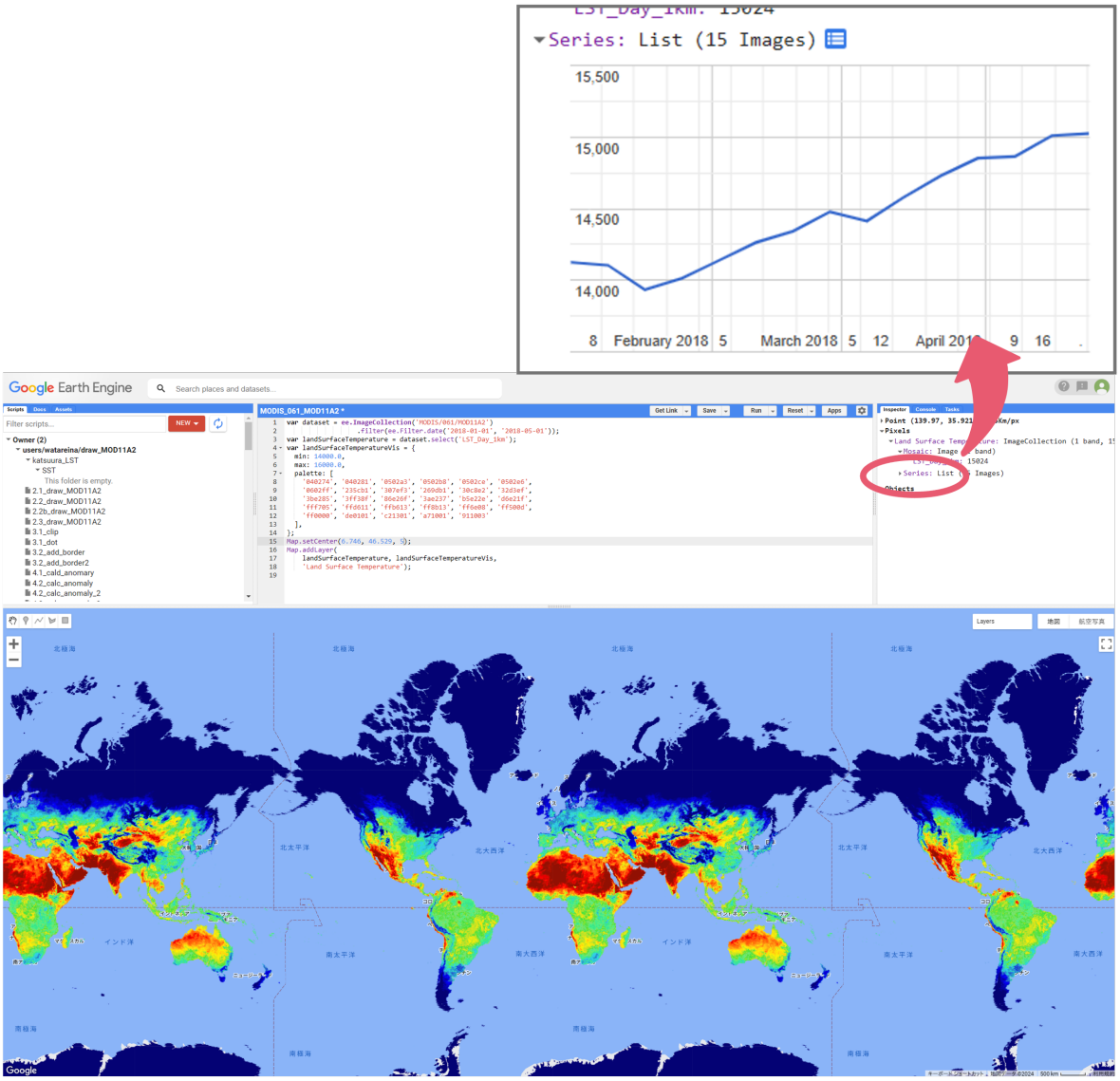

図 6．サンプルスクリプトの実行結果。上図は地図で指定したある地点における地表面温度の時系列変化

第 2 章

GEE スクリプトの説明と簡単な編集

第 1 章 1.4 でインポートしたサンプルスクリプトを個人のスクリプトファイルにコピーすることで、自由に編集・保存することができる。本章では、まずスクリプトファイルの作成方法とスクリプトに書かれてある命令文（プログラム）の中身について解説する。その後、調べたい期間のデータ抽出や単位の変換といった、数分程度でできる簡単な編集方法についても説明する。これらの超基本的な描画スキルを押さえておくだけでも、今後研究したいテーマの “下見” として十分に役立つだろう。

2.1. スクリプトファイルの作成

2.1.1. 新規スクリプトファイルの作成

Scripts -> New -> File でファイルを作成する *（ここでは名前を「draw_MOD11A2」とする）。すると、左のウィンドウの Script タブ内に「draw_MOD11A2」ファイルが追加される。これをクリックすると、中央のウィンドウが draw_MOD11A2 のスクリプト編集画面になる。第 1 章 1.4 でインポートしたスクリプトを、draw_MOD11A2 に保存してみよう。別のタブで「MOD11A2.061 Terra Land Surface Temperature and Emissivity 8...」のサンプルスクリプトを取得し、draw_MOD11A2 の編集画面に貼り付ける。編集画面右上の「Save」をクリックで保存完了である。なお、初めてファイルを作成する際には、ホームフォルダと Git リポジトリを作成する必要がある。これらの名前は変更できないので、慎重に検討の上作成しよう。

2.1.2. インポートしたスクリプトの説明

以下、draw_MOD11A2 にペーストされた “MOD11A2.061 Terra Land Surface Temperature and Emissivity Daily Global 1km” を読み込むスクリプトである。

```
1
2  var dataset = ee.ImageCollection('MODIS/061/MOD11A2') //検索データセット(MOD11A2)
3            .filter(ee.Filter.date('2018-01-01', '2018-05-01'));    //日にちの範囲を指定
4  var landSurfaceTemperature = dataset.select('LST_Day_1km'); //データの名称(図4のbandタブ)
5  var landSurfaceTemperatureVis = {
6      min: 14000.0, //色付けの際の最小値(*)
7      max: 16000.0, //色付けの際の最大値(*)
8      palette: [     //カラーパレット(色)
9            '040274', '040281', '0502a3', '0502b8', '0502ce', '0502e6',
10           '0602ff', '235cb1', '307ef3', '269db1', '30c8e2', '32d3ef',
11           '3be285', '3ff38f', '86e26f', '3ae237', 'b5e22e', 'd6e21f',
12           'fff705', 'ffd611', 'ffb613', 'ff8b13', 'ff6e08', 'ff500d',
13           'ff0000', 'de0101', 'c21301', 'a71001', '911003'
14     ],
15 };
```

次ページへ

```
15 Map.setCenter(6.746, 46.529, 2);    //地図表示の中心座標(経度, 緯度, ズームの大きさ[1–15])
16 Map.addLayer(   //描画
17           landSurfaceTemperature, landSurfaceTemperatureVis,
18           'Land Surface Temperature');
```

1–2 行目：dataset” という変数に、ある期間（ここでは 2018年 1 月 1 日 -2018年 4 月 30 日）の MOD11A2 データが代入されている。終わりが 05-01 となっているが、1 日前の 4 月 30 日までである。

3 行目：“landSurfaceTemperature” という変数に、“dataset” に含まれる 'LST_Day_1km' というバンド情報が代入されている。変数は任意なので、例えば “LST” などと定義しても OK。MOD11A2 プロダクトに格納されているバンドについては、図 4 で示したページの「Bands」タブで確認できる。「Bands」タブ内にある表の Name を見てみると、上から順に LST_Day_1km, QC_Day, Day_view_time, Day_view_angle, LST_Night_1km, QC_Night, …と並んでおり、計 12 種のバンド情報が格納されていることが分かる。その右側には Units(単位)、Min, Max(格納されている Digital Number (DN) 値の最大・最小値)、Scale(スケールファクタ)、Description(データ項目) が記載されている。DN 値とは、データ格納の都合上観測値を整数に変換した値のことである。今回使用する ‘LST_Day_1km’ は、単位がケルビン [K] で、7500–65535 の範囲の DN 値が格納された、Daytime Land Surface Temperature（日中の地表面温度）である。DN 値から Kelvin への変換はスケールファクタの 0.02 を掛ければよい。

4–14 行目：‘LST_Day_1km’ データの DN 値 (ここでは 14000–16000 の範囲) に対応する色を割り当て、その情報を変数 “landSurfaceTemperatureVis” に代入している。pallete 内の 6 桁はカラーコード（先頭から 2 桁毎に 16 進数で赤・緑・青の順に格納）。ここを編集すると任意の色を割り当てられるが、特にこだわりがなければ既定の配色で十分である。

15 行目：地図表示をする際の中心座標を設定している。ズームは大きいほど拡大表示される。

16–18 行目：“landSurfaceTemperature” を、“landSurfaceTemperatureVis” の配色設定で描画する。18 行目の 'Land Surface Temperature' は図のタイトルである。これは複数の図を表示する際などに便利。

2.2. 解析期間の設定

2.1 で解説したサンプルコード 2 行目の日付を編集すれば、任意の解析期間における LST データを抽出することができる。ただし、抽出された LST データそのまま Map.addLayer 関数に渡すと、日付の新しい LST が（雲などによる欠損部分を除いて）優先的に前面に表示される。もちろん、一つの観測周期分（Daily データなら 1 日分、8-day データなら 8 日分）の LST を見たい場合には問題ないが、例えば 1 ヶ月の代表値を見たい場合には、平均値や中央値といった何かしらの統計量に変換する必要があるだろう。

統計量の計算には **reduce 関数**を用いる。2023 年 7 月 16 日から 2023 年 7 月 31 日までの期間の LST データを抽出し、その中央値 (median) を求めたい場合は以下のように記述する。

```
1 var dataset = ee.ImageCollection('MODIS/061/MOD11A2’)
2           .filter(ee.Filter.date('2023-07-16','2023-08-01'));
3 var landSurfaceTemperature = dataset.select('LST_Day_1km’)
4           .reduce(ee.Reducer.median());
```

reduce 関数は、中央値「.reduce(ee.Reducer.median())」の他にも平均値「.reduce(ee.Reducer.mean())」、最大値「.reduce(ee.Reducer.max())」、最小値「.reduce(ee.Reducer.min())」、合計値「.reduce(ee.Reducer.sum())」、標準偏差「.reduce(ee.Reducer.stdDev())」などがある。さらに reduce 関数には線形回帰のオプション「ee.Reducer.linearRegression()」もある。

連続した数日間だけでなく数年間にわたる特定期間の LST データを抽出したい場合は、サンプルコードとは異なる方法で期間を設定する。例えば 2000 年から 2022 年の 23 年間を対象とする場合、これまで通り日付を編集して「.filter(ee.Filter.date('2000-07-01', '2022-08-01'))」とするだけでは、他の月のデータも一緒に抽出されてしまう。ここから更に 7 月後半だけを取り出すには、dataset と landSurfaceTemperature の変数をそれぞれ以下のように定義する。

```
var dataset_normal = ee.ImageCollection('MODIS/061/MOD11A2')
          .filterDate('2000-07-01','2022-08-01').filter(ee.Filter.dayOfYear(197,213));
var landSurfaceTemperature = dataset_normal.select('LST_Day_1km')
          .reduce(ee.Reducer.median());
```

「//」以下の文は実行結果には反映されず、コメント文として機能する。2 行目の「.filterDate('2000-07-01','2022-08-01')」で他の月も含めた 23 年分のデータを抽出した後、3 行目の「.filter(ee.Filter.dayOfYear(197, 213))」で 7 月後半のみを抽出している。197, 213 は、それぞれ 7 月 16 日と 8 月 1 日に相当する Day of Year (DOY) である（うるう年は 198, 214 になるが、ここでは考慮していない）。

2.3. スケールファクタの適用・単位変換

格納されている LST の値は、データ保存の都合上、整数の Digital Number (DN) 値となっている。これをケルビン[K]に換算するには、scale(offset)=0.02 を掛ければよい。さらにこれを、よりなじみの深い摂氏 [°C] に変換したい。 LST (摂氏) = DN × 0.02 − 273.15 であり、この計算は以下のスクリプトで実行できる。

```
//DN値→ケルビン→摂氏への変換
var LST_degC = landSurfaceTemperature.multiply(0.02).subtract(273.15);  //(DN*0.02)-273.15
var LST_degC_Vis = {
    min: 10,
    max: 50,
    palette: ['blue', 'green', 'yellow', 'orange', 'red'],
};
Map.setCenter(135, 35, 4);
Map.addLayer(LST_degC, LST_degC_Vis,'Land Surface Temperature');
```

2 行目で摂氏への変換が行われており、先ほど定義した “landSurfaceTemperature”（DN 値）に、「.multiply()」関数を用いて（）内の 0.02 を掛け、その後「.subtract()」関数で（）内の 273.15 を引いている。なお、掛け算と引き算以外の演算は他にも、足し算「.add()」、割り算「.divide()」、剰余「.mod()」などがある。摂氏の LST データを描画するので、描画色を割り当てる範囲（最小値と最大値）も 10–50 に変更した。

2000 年から 2022 年の 23 年間のデータを使って、7 月後半の LST の中央値 [°C] を算出してみよう。「Run」をクリックして以下の図 7 と同様のものが出力されれば成功である。

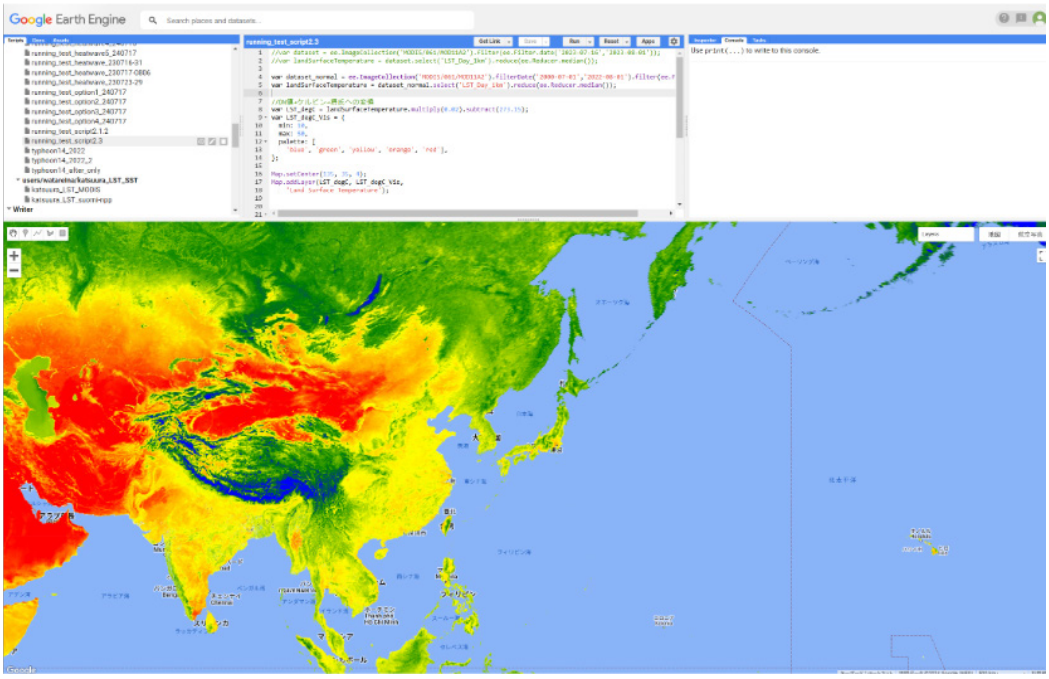

図 7．2000 年 –2022 年における 7 月後半の LST の中央値（℃）

第 3 章

アノマリ計算

本章では、2023 年 7 月後半に日本周辺で発生した猛暑を例に、衛星データを使った簡単な陸面解析を行ってみる。これまでに解説したスキルを多く使うので、よい復習になるだろう。

3.1. 複数の衛星プロダクトを読み込む

使用する衛星データは、MODIS センサの観測データから推定された LST データ「MOD11A2.061 Terra Land Surface Temperature and Emissivity 8-Day Global 1km」と植生指数（VI: Vegetation Index）データ「MOD13A2.061 Terra Vegetation Indices 16-Day Global 1km」である。データの検索方法やスクリプトの雛型を取得する方法については第 1 章を参照しよう。

新しいスクリプトファイル「calc_anomaly」を作成し、LST データと植生指数データを読み込むスクリプトを張り付けて以下のように整理する。

```
1  //LST, VIデータの読み込み
2  var LST_dataset = ee.ImageCollection('MODIS/061/MOD11A2')
3  var VI_dataset = ee.ImageCollection('MODIS/061/MOD13A2')
4
5  //日中のLSTデータとNDVIデータの読み込み
6  var LST = LST_dataset.select('LST_Day_1km');
7  var NDVI = VI_dataset.select('NDVI');
8
9  //描画色の設定（LST）
10 var LST_Vis = {
11     min: 14000.0,
12     max: 16000.0,
13     palette: [
14           '040274', '040281', '0502a3', '0502b8', '0502ce', '0502e6',
15           '0602ff', '235cb1', '307ef3', '269db1', '30c8e2', '32d3ef',
16           '3be285', '3ff38f', '86e26f', '3ae237', 'b5e22e', 'd6e21f',
17           'fff705', 'ffd611', 'ffb613', 'ff8b13', 'ff6e08', 'ff500d',
18           'ff0000', 'de0101', 'c21301', 'a71001', '911003'
19     ],
20 };
21
```

次ページへ

```
//描画色の設定（NDVI）
var NDVI_Vis = {
    min: 0,
    max: 9000,
    palette: [
        'ffffff', 'ce7e45', 'df923d', 'f1b555', 'fcd163', '99b718', '74a901',
        '66a000', '529400', '3e8601', '207401', '056201', '004c00', '023b01',
        '012e01', '011d01', '011301'
    ],
};

//描画
Map.setCenter(135, 37, 5);
Map.addLayer(LST, LST_Vis,'LST');
Map.addLayer(NDVI, NDVI_Vis,'NDVI');
```

データの読み込みに関しては、あとで異なる二つの期間（2023 年 7 月下旬とそれ以外の年の同時期）でフィルターをかけるため、サンプルスクリプトにあった ".filter()" の部分は取り除いた。長かった変数名も短くした。7 行目では VI_dataset から NDVI（Normalized Difference Vegetation Index: 正規化植生指標）データが抽出されている。データセットの説明（図 4 と同様のもの）を見ると、物理量データは NDVI と EVI(Enhanced Vegetation Index: 拡張植生指標）が格納されており、サンプルスクリプトでは NDVI が選択されている。どちらも植生の健康状態や密度を測るための指標であるが、NDVI の方が計算がシンプルで広く利用される。EVI は計算により多くの観測情報を使用するが、NDVI よりも背景土壌の影響を受けにくく、植生が密集している地域において感度が高いという特徴がある。

試しに「Run」でスクリプトを実行させてみよう。少々時間がかかるが、LST の分布が表示された後、NDVI の分布が上塗りされたことだろう。また、7 行目の ('NDVI') を ('EVI') に変更して実行すると、日本の植生域のコントラストが明瞭になったことが確認できる。

3.2. 平年値からの偏差の算出

2023 年 7 月後半の LST は平年と比べてどれだけ高くなっていたかを調べるためには、対象期間における LST の中央値と、複数年分にわたる同期間の LST から作成した平年値を算出し、それらの差分を求めればよい。植生指数に関しても同様の手順をとる。これらの解析は第 2 章で解説したスキルを使う。
calc_anomaly の 5–7 行目を以下のように編集する。

```
//日中のLSTデータとEVIデータの読み込み
var LST_2023 = LST_dataset.filter(ee.Filter.date('2023-07-16','2023-08-01'))
          .select('LST_Day_1km').reduce(ee.Reducer.median()); //2023年7/16-7/31のLST
```

次ページへ

```
var LST_normal = LST_dataset.filterDate('2000-07-01','2022-09-01')
          .filter(ee.Filter.dayOfYear(197,213))
          .select('LST_Day_1km') .reduce(ee.Reducer.median()); //LSTの平年値
var NDVI_2023 = VI_dataset.filter(ee.Filter.date('2023-07-16','2023-08-01'))
          .select('NDVI').reduce(ee.Reducer.median());//2023年7/16-7/31のNDVI
var NDVI_normal = VI_dataset.filterDate('2000-07-01','2022-09-01')
          .filter(ee.Filter.dayOfYear(197,213))
          .select('NDVI').reduce(ee.Reducer.median()); //NDVIの平年値
```

平年値は 2000 年から 2022 年までの 23 年分のデータを用いて算出した。また、これらは DN 値のままなので、LST は摂氏に、NDVI は本来の指数値に変換する。VI データの説明を見るとスケールファクタは 0.0001 であり、これを適用する。以下のスクリプトを追加する（// 日中の LST データと NDVI データの読み込み の後の 17 行目あたりに追加）。

```
//スケールファクタの適用・単位変換
var LST_2023_degC = LST_2023.multiply(0.02).subtract(273.15);
var LST_normal_degC = LST_normal.multiply(0.02).subtract(273.15);
var NDVI_2023 = NDVI_2023.multiply(0.0001);
var NDVI_normal = NDVI_2023.multiply(0.0001);
```

// 描画色の設定 の値の範囲を LST：10-50、NDVI：0-0.9 に変更し、// 描画 の Map.addLayer 内の変数名を更新して実行してみよう（例えば NDVI → NDVI_2023）。なお、複数の変数を表示しようとすると時間がかかるので、一方は // でコメントアウトするとよい。NDVI_2023 を描画した場合、図 8 のようになる。

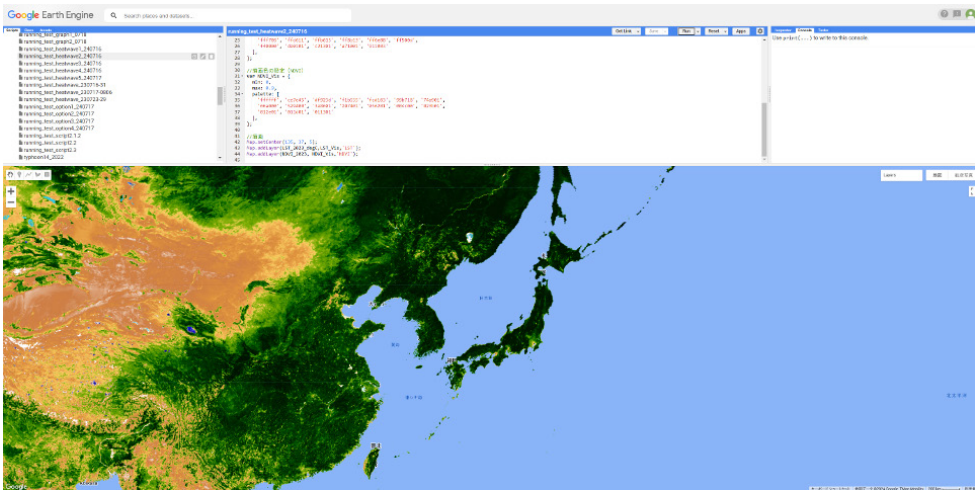

図 8．2023 年 7 月後半の NDVI 分布

次に、2023 年 7 月後半の LST から平年値を引いて偏差（Anomaly）を求める。引き算は 2.3 で行った摂氏換算のときと同様、「.subtract()」を用いる。スクリプトは以下の通り。

```
// 偏差（アノマリ）の算出
var anomaly_LST = LST_2023_degC.subtract(LST_normal_degC);
var anomaly_NDVI = NDVI_2023.subtract(NDVI_normal);
```

アノマリ用の描画色も設定する。今回は詳細なカラーコードを使わず、シンプルな 7 色にする。// 描画 の Map.addLayer() 内の変数を、アノマリの情報が代入されたものに変更する。

```
//アノマリの描画色の設定（LST）
var anomaly_LST_Vis = {
    min: -5.0,
    max: 5.0,
    palette: ['navy', 'blue', 'aqua', 'white', 'yellow', 'red', 'maroon'],
};

//アノマリの描画色の設定（NDVI）
var anomaly_NDVI_Vis = {
    min: -0.1,
    max: 0.1,
    palette: ['navy', 'blue', 'aqua', 'white', 'yellow', 'red', 'maroon'],
};

//描画
Map.setCenter(140, 37, 5);
Map.addLayer(anomaly_LST, anomaly_LST_Vis,'LST');
//Map.addLayer(anomaly_NDVI, anomaly_NDVI_Vis,'NDVI');
```

anomaly_NDVI をコメントアウトして anomaly_LST のみを描画した結果が図 9 である。日本は九州南部と四国を除いて昇温傾向がみられ、特に関東、東北、北海道の太平洋側の地域で＋ 5 ℃に達する昇温が起こっていたことが分かる。

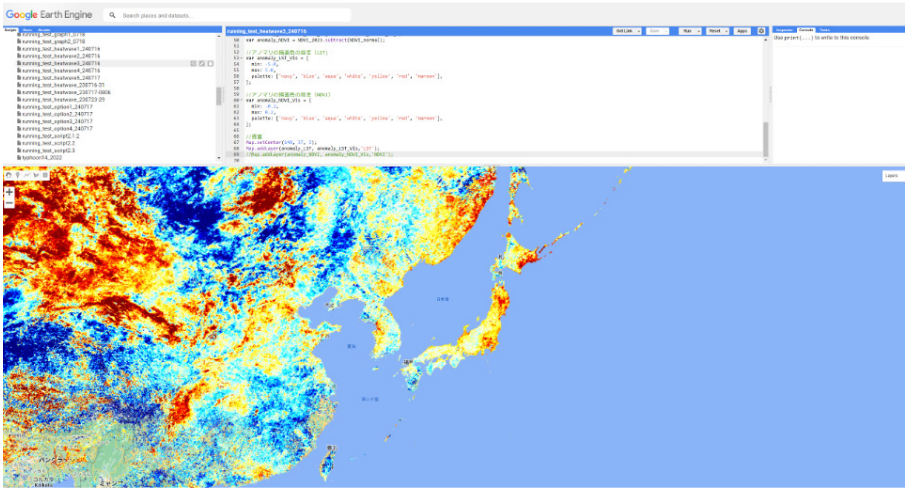

図 9．2023 年 7 月後半における LST のアノマリ分布（平年値からの差分）

ここでカラーバーの追加方法について触れておこう。ただし、GEE は描画した画像を高画質で出力することができない（第 5 章を参照）。そのため、描画結果を発表資料に用いる際には、別のデータ表示ソフトウェアで図の体裁を整える必要があり、GEE 上でカラーバー使う場面は少ないかもしれない。また、カラーバーを追加するためのスクリプトも非常に複雑である。「calc_anomaly」を複製して「calc_anomaly2」を作成し、試しに以下のスクリプトを末尾に追加して実行してみよう。

```
// カラーバーの追加
// カラーバーの位置設定
var legend = ui.Panel({
    style: {position: 'bottom-center', padding: '8px 15px'}
});

// カラーバーの表示形式設定
function makeColorBarParams(palette) {
    return {
        bbox: [0, 0, 1, 0.1],
        dimensions: '250x25',
        min: 0,
        max: 1,
        palette: palette
    };
}

// anomaly_LST_Visの描画色情報をカラーバーに反映
var colorBar = ui.Thumbnail({
    image: ee.Image.pixelLonLat().select(0),
    params: makeColorBarParams(anomaly_LST_Vis.palette),
});

// カラーバーの数値の設定
var legendLabels = ui.Panel({
    widgets: [
        ui.Label(anomaly_LST_Vis.min),
        ui.Label(
        ((anomaly_LST_Vis.max + anomaly_LST_Vis.min)/2),
        {textAlign: 'center', stretch: 'horizontal'}),
         ui.Label(anomaly_LST_Vis.max)],
    layout: ui.Panel.Layout.flow('horizontal')
});
// タイトルの設定
var legendTitle = ui.Label({
    value: 'LST Anomaly (degC)',
    style: {fontWeight: 'bold'}
});

// カラーバーを地図上に追加
var legendPanel = ui.Panel([legendTitle, colorBar, legendLabels]);
Map.add(legend.add(legendPanel));
```

図１０が出力結果である。カラーバーを表示させるためには、「位置の設定」「表示形式の設定」「描画色の反映」「数値・タイトルの追加」を別々に行い、最後にそれらを合わせて地図上に載せる。カラーバーの設定だけは、他のプログラミング言語と比べてずっと複雑である。GEE 上での使用はあまり想定されていないのかもしれない。

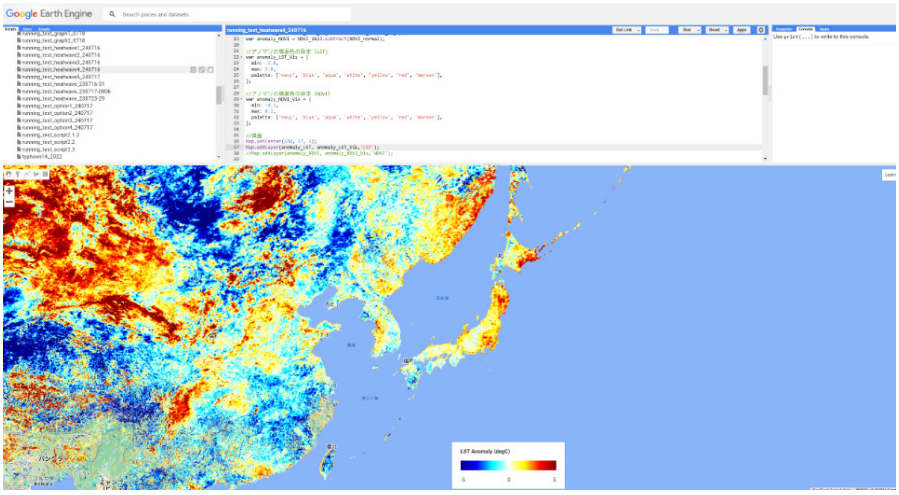

図１０．図９にカラーバーを追加したもの

3.3. 地図の並列表示

MapLinker という機能を使うと、複数の地図を並列表示できる。本章のような、LST の平年値とアノマリを並べて表示したい場合や、LST と EVI のそれぞれのアノマリを見比べたい場合などに便利である。Map のオブジェクトを新たに二つ作り、それぞれにレイヤー等を追加する。「calc_anomaly」ファイルの // 描画 以降のスクリプトを削除し、以下のスクリプトに差し替える。

```
//描画(map linker)
var map1 = ui.Map();
var map2 = ui.Map();
map1.setControlVisibility(true);
map2.setControlVisibility(true);

//mapに地図レイヤーを追加
map1.addLayer(anomaly_LST,anomaly_LST_Vis,'LST');
map2.addLayer(anomaly_NDVI,anomaly_NDVI_Vis,'NDVI');

//mapをまとめる配列を作成
var maps = [];
maps.push(map1);
maps.push(map2);
var linker = ui.Map.Linker(maps);
ui.root.widgets().reset(maps);

```

次ページへ

```
//mapの中央位置を設定
maps[0].setCenter(135,37,5);
```

全てを入力したら、「Save」＆「Run」で実行する。成功すれば、図１１のように並列表示された図（左が LST、右が NDVI）が表示される。NDVI の値が大きいほど植生の活性度が高いことを示す。つまり平年よりも NDVI が高い（暖色の）地域は植生の活性度が高く、NDVI が低い（寒色の）地域は活性度が低いことを意味している。例えば中国北東部に着目すると、LST アノマリの空間分布とは逆のコントラストが NDVI アノマリで見られる。平年よりも熱かった地域は植生の活性度は低かったようである。一方、日本では LST アノマリと NDVI アノマリとの間に逆の対応が見られない。日本の植生は猛暑が到来する前に梅雨を経験するため、熱さによるダメージはあまりなかったことがうかがえる。あるいは、1 km というやや粗い空間分解能では、農作物などへの影響を捉えられなかった可能性もあるだろう。

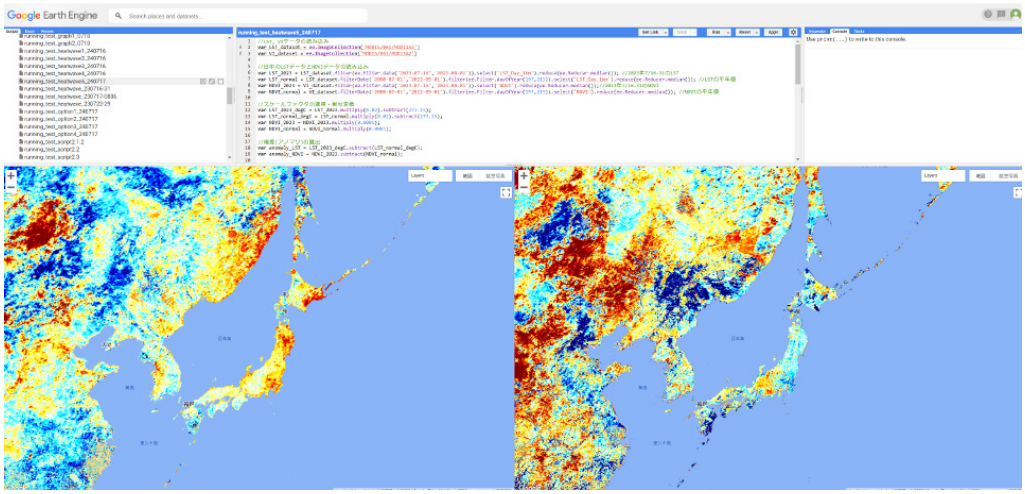

図１１．2023 年 7 月後半における LST（左）と NDVI（右）のアノマリ分布（平年値からの差分）

maps[0].setCenter() の () 内を北半球全域が表示されるように 70, 40, 3 に変更して「Run」してみよう。地中海、中央アジア、東アジアにかけて、LST のアノマリが正・負・正・負…と交互に分布している様子が確認できるだろうか？これにおおよそ対応するように、NDVI のアノマリも負・正・負・正…と反転して分布している。熱波と寒波は地球規模で影響を及ぼし合うことがある（テレコネクション）。図１２では、大規模な大気の高気圧・低気圧偏差の連なりが陸面環境に異常を起こした結果として、温度や植生活性度の正・負偏差の連なりが観測されている。

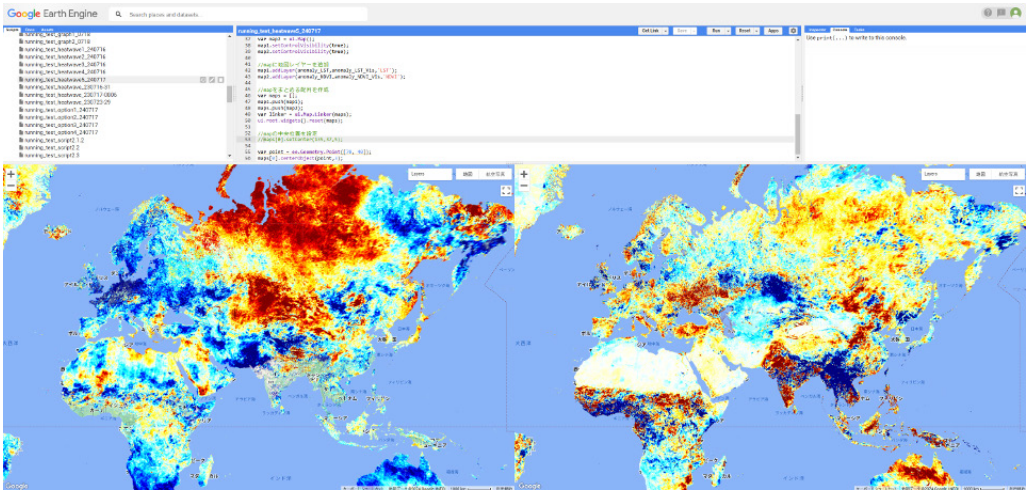

図１２．図１１の出力結果をユーラシア大陸で表示したもの

第 4 章

描画オプションの説明

この章では、地図上にドットや図形を描画する方法や、国境、政治境界、海岸線を描画する方法について解説する。これらの機能は、衛星データの描画結果で重要な箇所を強調させたり、国別・州別での結果比較をサポートしたりするほか、図として GEE 外にエクスポートする際の範囲設定にも便利である。

4.1. ドット・矩形領域の追加

ドットやライン、矩形、その他の図形を描画するには、図 7 左上の各種ボタンからマウスドラッグで描く方法が最も手軽な方法である。ただし、任意の緯度経度座標を正確に反映させて描きたい場合には、Geometry 関数を用いてコマンド上で操作する方法が便利である。

任意の緯度経度座標にドットを追加する方法は以下の通り。「〇〇〇（任意）」という新しいファイルを作成し、富士山付近の 138.7˚N, 35.36˚N に追加してみる。

```
1 //位置情報の定義(富士山付近)
2 var point = ee.Geometry.Point([138.7, 35.36]);
3 Map.addLayer(point);
```

矩形を描画する方法は以下の通り。139˚E–141˚E, 34.5˚N–36.5˚N の矩形を描画してみる。

```
1 //矩形の位置情報を定義　(139˚E–141˚E, 34.5˚N–36.5˚N)の矩形
2 var rectangle = ee.Geometry.Rectangle([139, 34.5, 141, 36.5]);
3 Map.addLayer(rectangle);
```

多角形も描画できる。(136˚E,34.5˚N), (137˚E,34.5˚N), (137.5˚E,35.3˚N), (136.5˚E,36˚N), (135.5˚E,35.3˚N) の位置情報をもつ五角形を描画してみる。

```
1 //五角形の位置情報を定義
2 //(136˚E,34.5˚N)から反時計回りに(137˚E,34.5˚N),(137.5˚E,35.3˚N),(136.5˚E,36˚N),(135.5˚E,35.3˚N)
3 var polygon = ee.Geometry.Polygon([[136,34.5],[137,34.5],[137.5,35.3],[136.5,36],[135.5,35.3]]);
4 Map.addLayer(polygon);
```

新規スクリプトファイル「add_geometry」を作成し、上記のコードを実行してみよう。地図の中心座標とスケールを「Map.setCenter(138.5, 35.5, 8);」とすれば、図１３のように出力される。

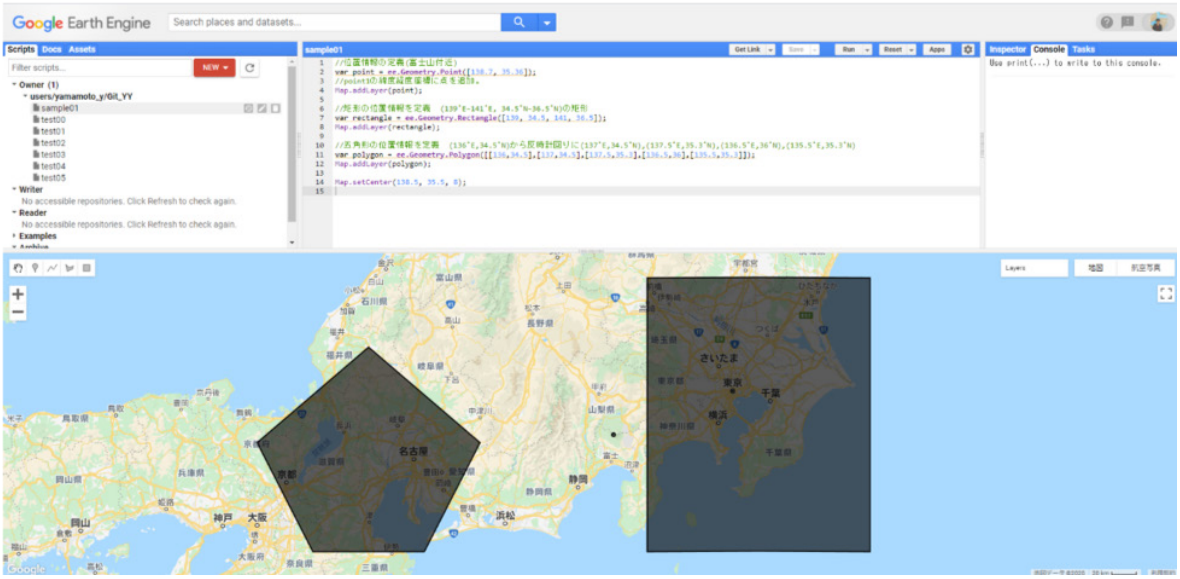

図１３．ドット、矩形、多角形の出力例

また、clip 関数を用いて衛星データを任意の図形領域で切り出すことも可能である。2.3 で算出した７月の LST を、139˚E–141˚E, 34.5˚N–36.5˚N の矩形領域で切り出して描画してみる。「draw_MOD11A2」ファイルの Map.setCenter と Map.addLayer の２文を消去あるいは「//」でコメントアウトし、下記のスクリプトで上書きして実行してみよう。

```
1 //矩形領域の設定（139˚E–141˚E, 34.5˚N–36.5˚N）
2 var rectangle = ee.Geometry.Rectangle([139, 34.5, 141, 36.5]);
3 //LST_degCを矩形領域で切り出して描画
4 var clip_image = ee.Image(LST_degC).clip(rectangle);
5 Map.setCenter(140, 35.5, 8);
6 Map.addLayer(clip_image, LST_degC_Vis);
```

すると図１４のように出力される。

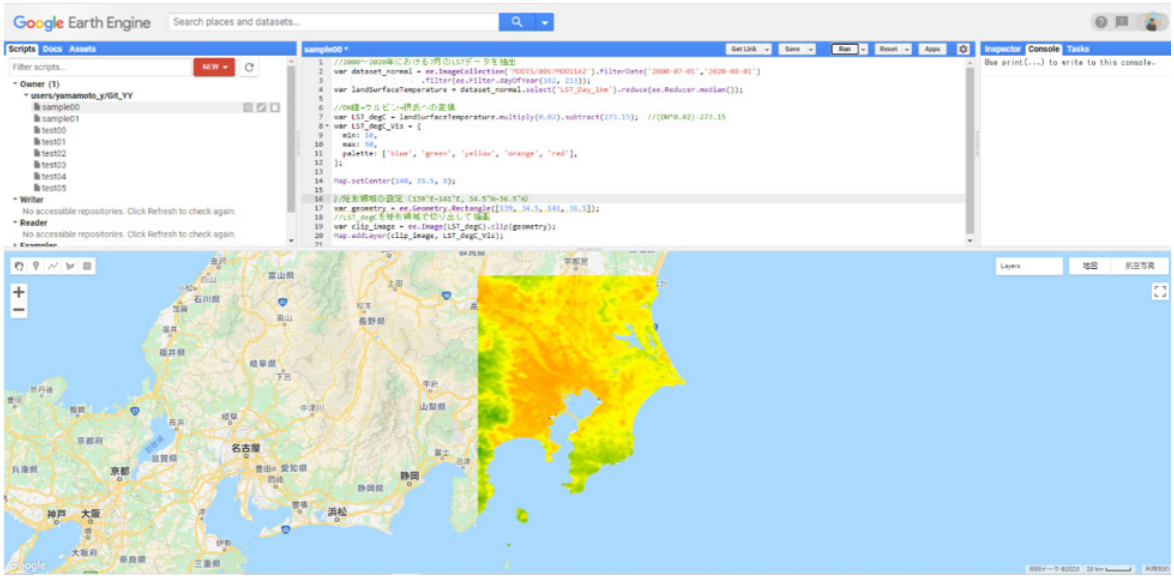

図１４．矩形領域で切り出された LST 分布

4.2. 海岸線・国境・行政境界の追加

海岸線と国境情報を読み込んで描画すると、データ表示をした際に位置のイメージがつかみやすくなり便利である。

```
//海岸線・国境情報の読み込み
var coastline = ee.FeatureCollection("USDOS/LSIB_SIMPLE/2017");
//海岸線・国境の描画色
var lineVis = {
    fillColor: 'FF000030', //塗り
    color: 'blue', //線
    width: 1.0 //線の太さ
};
var line = coastline.style(lineVis);
// 海岸線・国境を地図に追加。
Map.addLayer(line, {}, "USDOS/LSIB_SIMPLE/2017");
```

5 行目の fillColor のカラーコードが 8 桁になっている。これは 6 桁のカラーコード + 2 桁の不透明度(16 進数)で構成されている。今回は塗りを 30% にしているが、衛星データと合わせて描画する際などケースに応じて使い分けるとよい。新しいスクリプトファイル「add_border」を作成して実行してみよう。地図をズームアウトすれば、以下の図１５のように国境線が反映されたことが確認できる。

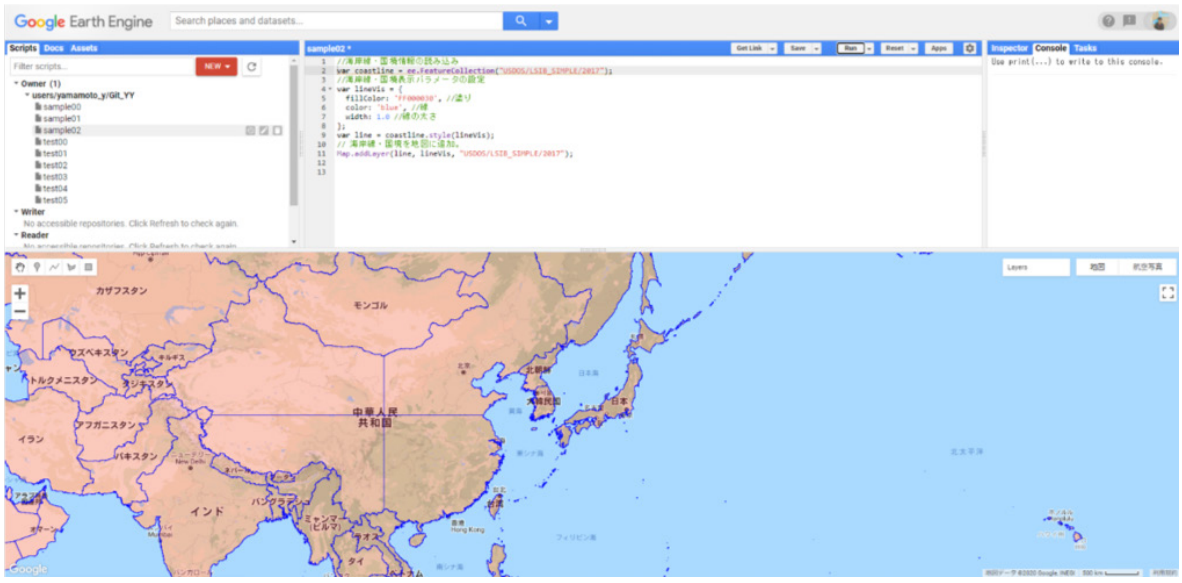

図１５．海岸線・国境の描画例

行政境界を描画するには、まず対象とする国の行政区分情報を取得した後、そこから目当ての行政境界情報を抽出するという手順をとる。今回はロシアのタイミル自治管区と日本の北海道の境界を描画してみる。新しいスクリプトファイル「add_border2」を作成し、以下のスクリプトを入力し、実行してみよう。

```
1  //対象とする国の行政区分情報を選択
2  var country = ee.FeatureCollection("FAO/GAUL/2015/level1"); //各国データ（国・行政区分あり）
3  var russia = country.filter(ee.Filter.eq('ADM0_CODE',204)); //ロシア(204 → 'Russian Federation')
4  var japan = country.filter(ee.Filter.eq('ADM0_CODE',126)); //日本（126 → 'Japan' でも可）
5
6  //該当地域を探す
7  print(russia);
8  print(japan);
9
10 //行政区分の描画色
11 var lineVis = {
12     fillColor: 'FF000030', //塗り
13     color: 'blue', //線
14     width: 0.5 //線の太さ
15 };
16
17 //描画
18 //タイミル自治管区
19 var taymyria = russia.filter(ee.Filter.eq("ADM1_NAME","Taymyrskiy Okrug"));
20 var taymyrialine = taymyria.style(lineVis);
21 Map.addLayer(taymyrialine);
22 //北海道
23 var hokkaido = japan.filter(ee.Filter.eq("ADM1_NAME","Hokkaidoo"));
24 var hokkaidoline = hokkaido.style(lineVis);
25 Map.addLayer(hokkaidoline);
26
27 Map.setCenter(115, 60, 3);
```

1–4 行目　：対象とする国の行政区分情報を取得している。まず該当の国に割り当てられた ADM0_CODE を（https://www.fao.org/nocs/en/）から探す。サイトページの一番右側のカラムに書かれている番号が ADM0_CODE である。ロシアは 204、日本は 126。あるいは ADM0_NAMEとして、204 を 'Russian Federation'、126 を 'Japan' としてもよい。

6–8 行目　：取得したロシアと日本の行政区分情報を print() 関数で出力する。出力結果は右のコンソール画面に表示される。print(japan) だと「FeatureCollection FAO/GAUL/2015/level1 (47 elements, 11 columns)」と出力される。ここの “features” をクリックして展開すると、47 都道府県のフィーチャ（図形・属性情報）が格納されていることが分かる。ひとつひとつの “Feature” を展開すると、複数の項目が格納されており、その中の “properties” をさらに展開する（図１６の右上）。すると、その中に「ADM1_NAME」という項目がある。ここの名前を確認し、該当地域を探す。北海道の ADM1_NAME は “Hokkaidoo” であり、これをもとに行政境界情報を抽出する（22 行目）。

10–15 行目：行政区分を描画する際の配色設定。海岸線と国境の描画の際に解説したので省略。

18–21 行目：6 行目で取得した ADM1_NAME「Taymyrskiy Okrug」をもとに、変数 “russia” からタイミル自治
：管区の行政境界情報を抽出し、描画している。

22–25 行目　18–21 行目と同様の方法で、北海道域を描画。

27 行目　：地図表示の中央位置の設定。

上記スクリプトの実行結果として、図１６と同様のものが描画されていれば OK である。

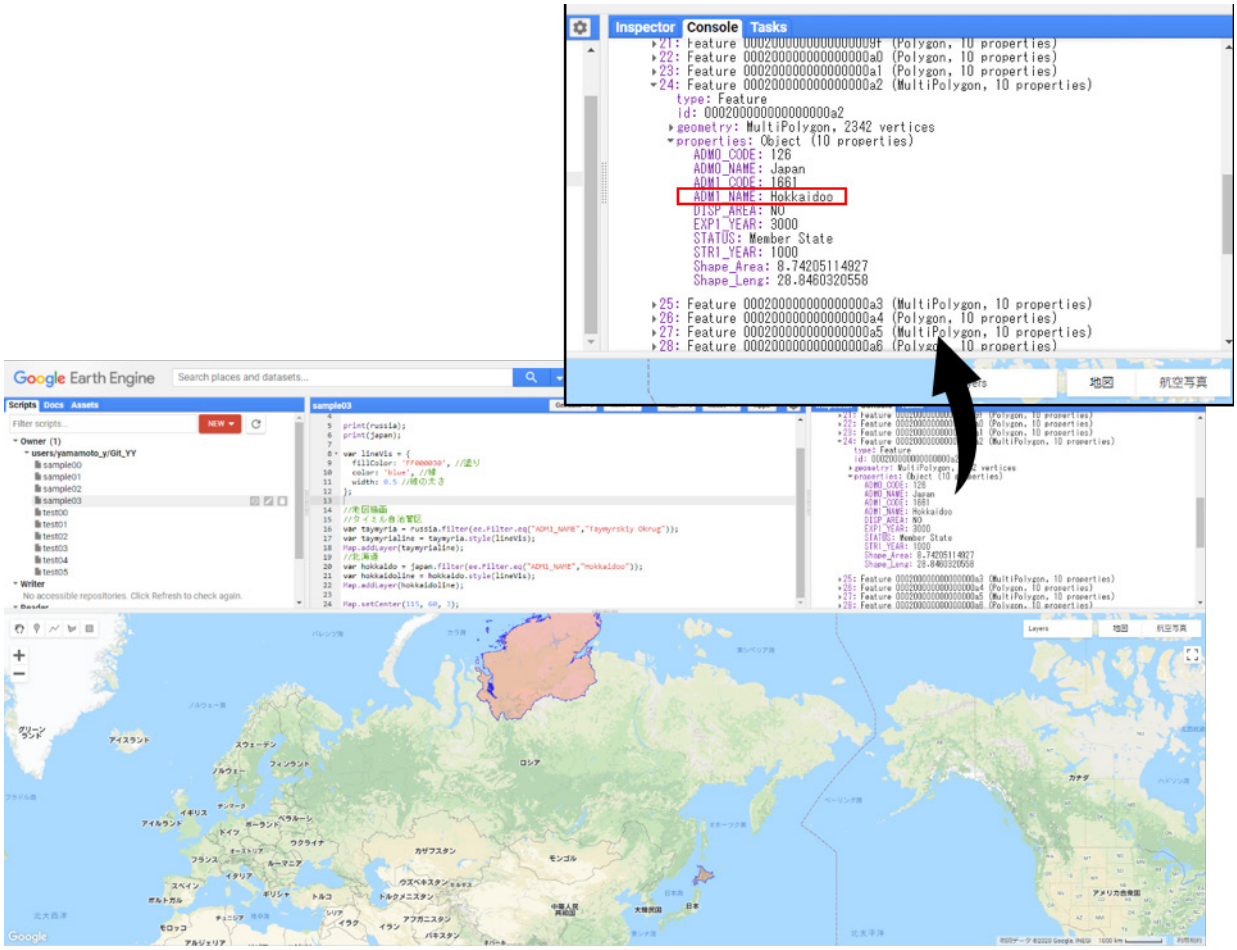

図１６．行政境界の追加例（ロシアのタイミル自治管区と日本の北海道）

第５章

作成した図のエクスポート

GEE は描画した図を GEE 外に保存できる。ただしローカル PC に直接出力することはできず、クラウドストレージ上に出力することになる。出力先は「Google Drive」・「Google Storage Cloud」・「Google Earth Engine Assets」からいずれかを選択する。今回は多くのユーザが選択するであろう「Google Drive」への出力方法について解説する。なお、出力データの形式は「GeoTIFF」か「TFRecord」を選択する。いずれも Windows の標準ソフトで開くことはできないが、GeoTIFF ファイルは GIS（Geographic Information System：地理情報システム）ソフトなどで容易に編集することができ、TFRecord ファイルは TensorFlow という Google が開発した Python ライブラリで扱うことができる。今回は広く使われている無料の GIS ソフト「QGIS」を用いて出力した GeoTIFF ファイルを編集し、別の画像ファイル形式として保存する方法について解説する。

5.1. Google Drive へ図をエクスポート

第 2 章 2.3 で算出した 7 月後半の LST を 139˚E–141˚E, 34.5˚N–36.5˚N の矩形領域で切り出し、図としてエクスポートしてみる。LST 分布を矩形領域で切り出すところまでは第 4 章 4.1 で既に行っており（図１４）、まずは新規ファイル「export_image」にそのスクリプトを追加する。「draw_MOD11A2」をペーストし、最後の 2 行の描画に関するコードを、第 4 章 4.1 の clip 関数を用いたスクリプトに差し替えればよい。

```
//2000-2022年における7月後半のLSTデータを抽出
var dataset_normal = ee.ImageCollection('MODIS/061/MOD11A2')
                     .filterDate('2000-07-01', '2022-09-01')
                     .filter(ee.Filter.dayOfYear(197,213));
var landSurfaceTemperature = dataset_normal.select('LST_Day_1km')
                     .reduce(ee.Reducer.median());
//DN値→ケルビン→摂氏への変換
var LST_degC = landSurfaceTemperature.multiply(0.02).subtract(273.15);
var LST_degC_Vis = {
    min: 10,
    max: 50,
    palette: [
          'blue', 'green', 'yellow', 'orange', 'red'],
};

//矩形領域の設定（139˚E–141˚E, 34.5˚N–36.5˚N）
var rectangle = ee.Geometry.Rectangle([139,34.5,141,36.5]);
//LST_degCを矩形領域で切り出して描画
var clip_image = ee.Image(LST_degC).clip(rectangle);
```

```
Map.setCenter(140, 35.5, 8);
Map.addLayer(clip_image, LST_degC_Vis
```

切り出された LST データは変数 ”clip_image” に代入されている。次に Export.image.toDrive() 関数を用いて ”clip_image” を Google Drive にエクスポートする。「export_image」の末尾に以下のコードを追加する。

```
//画像をgoogleドライブに出力する
Export.image.toDrive({
          image: clip_image,
          description: 'test_image',
          scale: 1000,
          region: rectangle,
          maxPixels: 1e13
});
```

「image: 衛星データが代入された変数」、「description: 出力ファイル名」、「scale: 空間解像度 [m]」、「region: 出力範囲が代入された変数」、「maxPixels: 出力画素数の上限」をそれぞれ指定する。空間解像度はデータセットの説明（図４）の Bands タブの最上部に記載されている。今回扱った LST データは 1000 m である。maxPixels は出力する画素数が非常に大きい場合に上限数を指定する。指定しない場合のデフォルトの上限画素数は 1000000 であり、今回出力する画素数はそれ以下なので無くてもよい。

「Run」をクリックして実行すると、右のウィンドウの「Tasks」というタブが橙色で表示される（図１７左）。Tasks タブ内の「Run」をクリックすると図１７右のようなウィンドウが表示される。スクリプト上で設定した空間分解能・出力先のクラウドストレージ・ファイル名が正しく反映されているか確認する。今回は Google Drive 内の GEE フォルダを出力先に指定したいので、「Drive folder」欄に GEE と入力する。指定したフォルダが存在しない場合は新しく作成される。出力設定が完了したら下部の「Run」をクリックする。ファイル作成まで数分ほどかかる場合があるが、指定したフォルダに ”test_image.tif” が出力されているはずである。

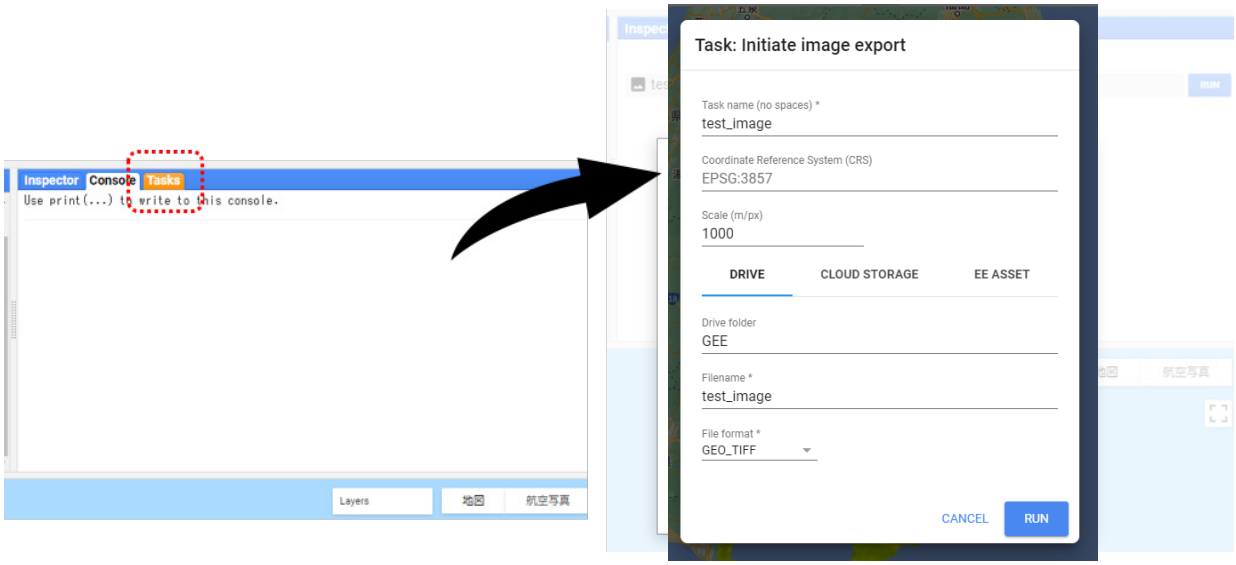

図１７．「Tasks」タブ内の画像のエクスポートに関するウィンドウ

5.2. QGIS に描画・任意の画像形式で保存

QGIS の HP（https://www.qgis.org/download/）より各自の OS に合うインストーラをダウンロードし、インストールする。

QGIS を起動して新規プロジェクトを開き、プロジェクトウィンドウ（図１８の赤枠で示された場所）に test_image.tif をドラッグ＆ドロップする。すると図１８のように LST 分布がグレースケールで表示される。

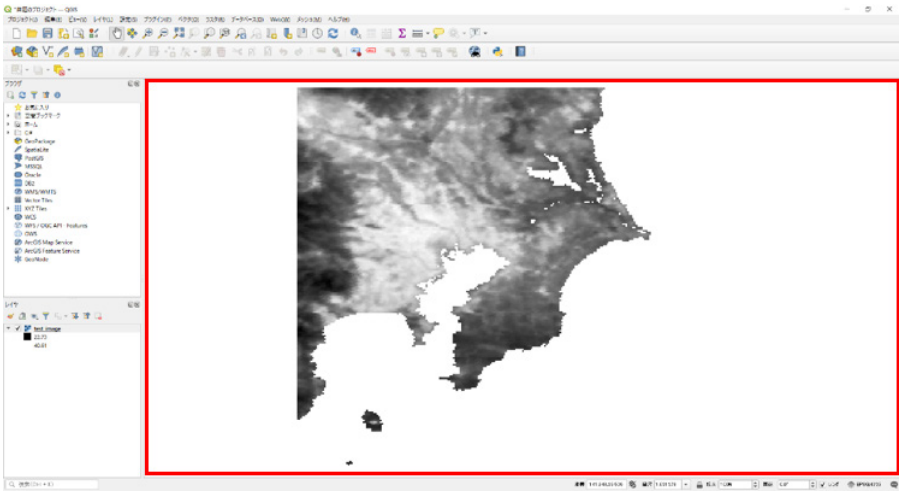

図１８．test_image.tif を QGIS 上で開いた際の画面

グレースケール以外の配色を指定することもできる。左下のウィンドウにあるファイル名 “test_image” を右クリックし、プロパティを開く。すると図１９左のようなウィンドウが現れる。"シンボロジ"タブでレンダリングタイプを単バンドグレーから単バンド疑似カラーに変更し、好みのカラーランプを選択する（図１９中央）。カラーランプは自由にカスタマイズすることができる。ここでは特にこだわらず Spectral を選択した。低い値に暖色が割り当てられているので、カラーランプリスト上部の 「カラーランプを反転」にチェックを入れて OK または適用をクリックする。すると図１９右のようにカラーの LST 分布が表示される。

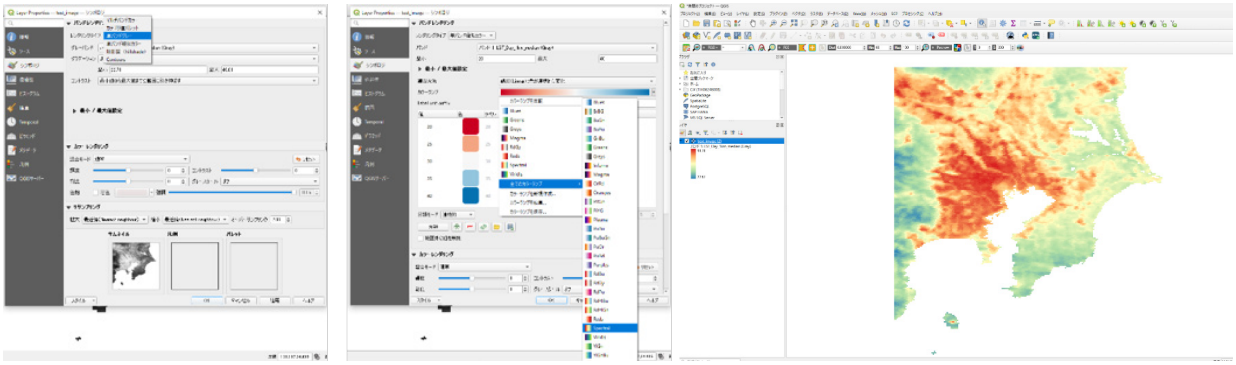

図１９．QGIS 上での描画色の編集画面

LST 分布を画像として保存したい場合は、左上の「プロジェクト」タブから「インポートとエクスポート」→「地図を画像にエクスポート」を選択し、画像の大きさや dpi、ファイル形式を指定して出力する。
GeoTIFF 以外の形式で生データを出力したい場合は、左下ウィンドウ内の “test_image” を右クリックして「エクスポート」→「名前を付けて保存」を選択し、ファイル形式を指定して出力する。

第 6 章

時系列グラフの作成

GEE は簡単なグラフを描くこともできる。グラフは Console に出力され、マウスオーバーで各点の値も確認できるほか、グラフを PNG や SVG 形式で出力したり、グラフデータを CSV 形式で出力したりできる。今回も 2023 年 7 月の猛暑を例として解説していく。第 3 章の解析結果から、猛暑による地表面の昇温は関東地方で顕著であったことが分かった。そこで解析対象範囲を関東周辺に絞り、「2023 年 7 月に LST がどのように変化したか」、「その変化は他の年とどれだけ異なっていたか」をグラフ化する。

6.1. 時系列データ・解析範囲とするポリゴンの作成

まずは新規のスクリプトファイル「draw_timeseries」を作成する。“MOD11A2.006 Terra Land Surface Temperature and Emissivity 8-Day Global 1km” プロダクトから、2014 年-2023 年における 7 月の LST データを取得する（取得方法は第 2 章 2.2 を参照）。

```
//2014年-2023年における7月のLSTデータを抽出
var dataset = ee.ImageCollection('MODIS/061/MOD11A2')
          .filterDate('2014-06-01','2023-09-01').filter(ee.Filter.dayOfYear(183,213));
var LST = dataset.select('LST_Day_1km');});
```

変数 “LST” 内の DN 値を摂氏に変換する。第 2 章 2.3 では、LSTの期間データを統計量として一つにまとめてから「.multiply(0.02).subtract(273.15)」を適用して変換した。今回は時系列変化の情報を失いたくないので、統計量としてまとめずに摂氏に変換する。ただし、直接「LST.multiply(0.02).subtract(273.15)」とするとエラーを起こしてしまうので別の方法で変換する。スクリプトは以下の通り。

```
//DN値→ケルビン→摂氏への変換
LST = LST.map(function(image) {
 return image.addBands(image.multiply(0.02).subtract(273.15).rename('Daytime_LST'));
});
var LST_degC = LST.select('Daytime_LST');
```

「map(function(image){})」内で計算させることで、各時間の LST データに対して演算処理を行うことができる。「image.addBands」は新たなバンドを追加する関数である。ここでは変数 “LST” に追加された新しいバンド ‘Daytime_LST’ に、摂氏に変換された LST データが格納されている。ちなみに、「image.select().addBands」とすると、DN 値が格納されたバンド 'LST_Day_1km' を、摂氏の LST が格納されたバンド ‘Daytime_LST’ に上書きできる。

次に解析領域を定義するためのポリゴンを作成する。第 4 章 4.1 では Geometry 関数を用いて作成していたが、今回はマップからマウスドラッグで作成してみる。マップの左上の描画オプションから「図形を描画」を選択し、任意のポリゴンを作成する。本解析では朝鮮半島全域を囲うような形にした（図 20）。その後「Exit」ボタンをクリックすると、スクリプト編集画面の上部にポリゴンの位置情報を格納した以下のスクリプトが作成される。デフォルトの変数名は “geometry” であるが、各自好きな名前に変更できる。

```
Import (1 entry)
var geometry: Polygon, 4(or 5) vertices
```

図２０．マウスドラッグで作成されたポリゴンの例

6.2. グラフの描画とデータダウンロード

まずは 2023 年 7 月における LST の時系列変化を描画する。連続する一期間を描画する最もシンプルなケースであり、ui.Chart.image.series() 関数が用いられる。

```
//2023年7月のLST
var LST_degC_2023 = LST.select('Daytime_LST').filterDate('2023-07-01','2023-08-01');

//時系列グラフの描画
var series1 = ui.Chart.image.series(LST_degC_2023,geometry5,ee.Reducer.mean(),1000);
print(series1); //2023年のみ
```

5 行目に記載されているように、ui.Chart.image.series() 関数に必要な情報は、(LST データの変数), (ポリゴンの変数), (Reducer), (空間分解能(m)) である。print() にグラフの変数を渡すことで、右側のコンソール画面にグラフが表示される（図２１）。カーソルを当てるとグラフ内の点の値を確認できる。

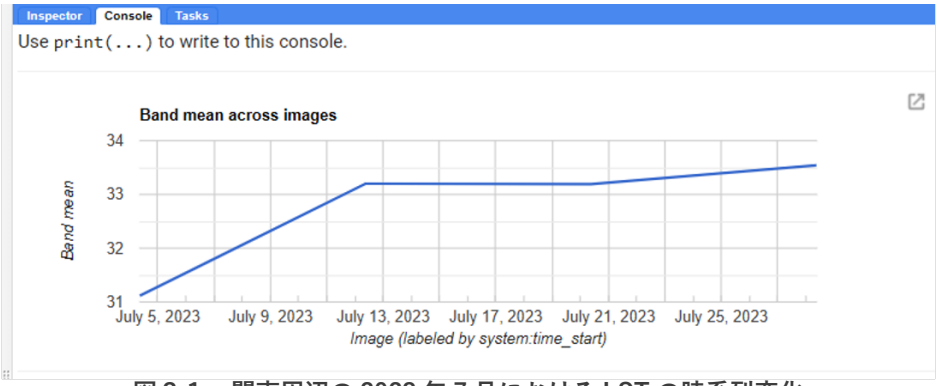

図２１．関東周辺の 2023 年７月における LST の時系列変化

次に、図２１で示した 2023 年の LST の時系列変化を他の年のものと比べてみる。同一期間で年ごとの比較を行う場合には ui.Chart.image.doySeriesByYear() 関数が有効である。

```
//経年変化グラフの描画
var series2 = ui.Chart.image.doySeriesByYear(
          LST_degC,'Daytime_LST',geometry5,ee.Reducer.mean(),1000);
print(series2); //2014年-2023年
```

ui.Chart.image.doySeriesByYear() 関数に必要な情報は、(LST データの変数), (バンド名), (ポリゴンの変数), (Reducer), (空間分解能 [m]) である。出力結果は図２２。関東では、2023 年の夏がここ数年で最も暑かったことが見て取れる。

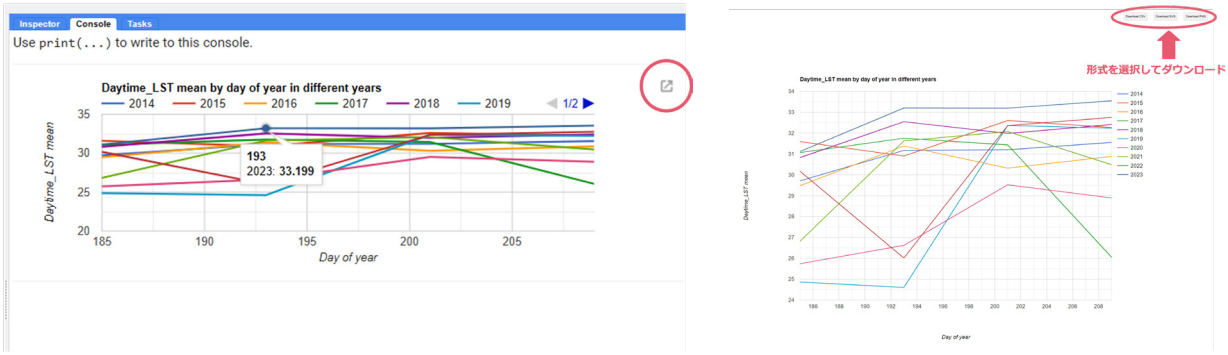

図２２．関東周辺の７月における LST の時系列変化（2014 年～ 2023 年）

グラフ右上にある四角矢印をクリックして拡大すると、CSV, SVG, PNG 形式のいずれかを選択してダウンロードすることができる。

今回は、GEE を利用した衛星データの描画方法や簡単な演算処理、時系列プロットの作成方法について学んだ。第１章の冒頭でも述べたとおり、衛星データは様々な分野で高い需要がありながら、手軽には利用しにくいのかもしれない。本講習会では、GEE を利用すれば、広域を対象とした衛星データの解析が容易にできることを実感できたと思う。そしていろいろな衛星データに手軽に触れていくことで、今後の研究の新たな視点となれば幸いである。