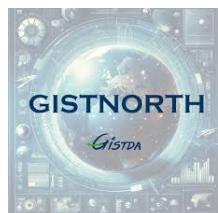


คู่มืออบรมเชิงปฏิบัติการ

การใช้ Google Earth Engine เพื่อสนับสนุนการบริหารงานท้องถิน



ศักดิ์ดา ห้อมหาล และ ชวิศ ศรีเมณี

ศูนย์ภูมิภาคเทคโนโลยีอวกาศและภูมิสารสนเทศ (ภาคเหนือ) (GISTNORTH)

ภาควิชาภูมิศาสตร์ คณะสังคมศาสตร์ มหาวิทยาลัยเชียงใหม่

2568

คำนำ

ปัจจุบันเทคโนโลยีภูมิสารสนเทศและข้อมูลจากดาวเทียมเข้ามามีบทบาทสำคัญต่อการตัดสินใจด้านนโยบาย และการบริหารจัดการทรัพยากรธรรมชาติและสิ่งแวดล้อมของท้องถิ่น การนำเครื่องมือที่สามารถประมวลผล ข้อมูลขนาดใหญ่ (Big Data) ได้อย่างรวดเร็วและแม่นยำมาประยุกต์ใช้จึงเป็นสิ่งจำเป็นอย่างยิ่ง ศูนย์ภูมิภาค เทคโนโลยีอิทธิพลภูมิสารสนเทศ (ภาคเหนือ) ภาควิชาภูมิศาสตร์ คณะสังคมศาสตร์ มหาวิทยาลัยเชียงใหม่ tron หนึ่งในผู้นำด้านความสำคัญดังกล่าวจึงได้จัดทำคู่มือการใช้งาน Google Earth Engine (GEE) ที่ช่วยให้นักวิจัย ผู้ปฏิบัติงานภาครัฐ และหน่วยงานท้องถิ่น สามารถวิเคราะห์ข้อมูลภาพถ่ายดาวเทียมและข้อมูลเชิงพื้นที่ ปริมาณมหาศาลได้ในเวลาอันสั้น

เนื้อหาในเล่มจะเริ่มตั้งแต่หลักการพื้นฐานของ Google Earth Engine (GEE) ทั้งส่วนของสถาปัตยกรรมระบบ การเข้าถึงข้อมูลดาวเทียม (เช่น Landsat, Sentinel, MODIS) รวมถึงวิธีการใช้งานอินเตอร์เฟซ Code Editor และ API ในภาษา JavaScript ตลอดจนตัวอย่างการประยุกต์ใช้จริงในสถานการณ์ต่างๆ อาทิ การติดตามการเปลี่ยนแปลงของการใช้ประโยชน์ที่ดิน และการวิเคราะห์ความเสี่ยงภัยพิบัติ นอกจากนี้ยังมีแนวทางการปรับแต่งโค้ดและสร้างแอ��พลิเคชันสำหรับนำเสนอข้อมูลให้กับผู้บริหารท้องถิ่นและประชาชนทั่วไป

หวังเป็นอย่างยิ่งว่าเอกสารเล่มนี้จะเป็นแรงบันดาลใจให้กับผู้ปฏิบัติงานด้านการบริหารท้องถิ่น และนักพัฒนาระบบ GIS ที่ต้องการใช้ประโยชน์จากข้อมูลภูมิสารสนเทศอย่างเต็มที่กัยภาพ

ท้ายที่สุด ผู้เขียนขอขอบคุณทีมพัฒนา Google Earth Engine และชุมชนนักพัฒนาทั่วโลก ที่แบ่งปันความรู้ และเครื่องมืออันทันสมัย ตลอดจนผู้ที่มีส่วนร่วมที่ทำให้เอกสารเล่มนี้บรรลุเป้าหมายในการเป็นคู่มือการใช้งาน GEE อย่างครบถ้วนสมบูรณ์

ผู้เขียน

พฤษภาคม 2568

สารบัญ

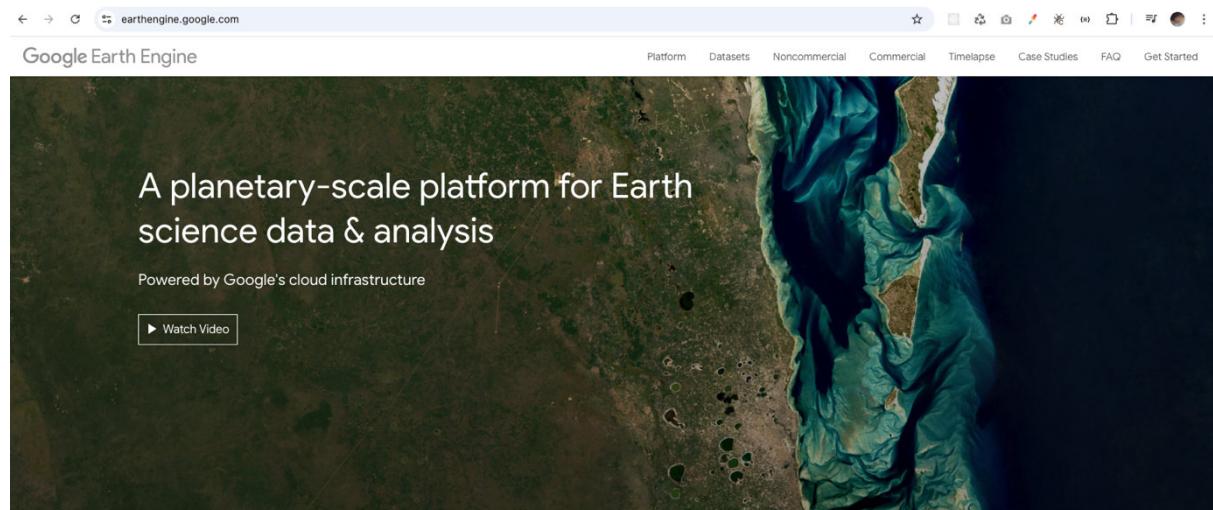
บทที่ 1: รู้จักกับ Google Earth Engine (GEE).....	1
1.1 Google Earth Engine คืออะไร	1
1.2 ความเป็นมา.....	1
1.3 ข้อดีและจุดเด่นของ GEE	2
1.4 ความสำคัญของ GEE ในการบริหารงานท้องถิ่น (Importance of GEE in Local Administration).....	3
บทที่ 2: การใช้งาน GEE Code Editor.....	7
2.1 การสมัครใช้งานและการเข้าสู่ GEE Code Editor.....	7
2.2 ส่วนประกอบต่างๆ ของ GEE Code Editor	17
บทที่ 3 ข้อมูลและแหล่งข้อมูลใน GEE	21
3.1 Imagery Dataset.....	21
3.2 Terrain / Elevation Dataset	23
3.3 Climate Dataset	24
3.4 Weather Dataset.....	25
3.5 Atmospheric Dataset	26
3.6 Land Cover / Land Use Dataset	27
3.7 Vegetation Indices Dataset.....	28
3.8 Hydrology Dataset.....	28
3.9 Soils Dataset	28
3.10 Fire.....	28
3.11 Nighttime Lights.....	28
3.12 Socioeconomic Dataset.....	29
3.13 Oceanography Dataset.....	29
3.14 Infrastructure Dataset.....	29
บทที่ 4 พื้นฐานภาษา JavaScript สำหรับ GEE.....	30
4.1 Server-side vs Client-side	30
4.2 ตัวแปร (Variables).....	32
4.3 ประเภทข้อมูล (Data Types)	32
4.4 การ Comments.....	33
4.5 พังก์ชัน (Functions)	34
4.6 การใช้ if	35
4.7 การใช้งาน Loop	36
4.8 แนวคิดการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming: OOP).....	37
บทที่ 5 GEE object	43
5.1 Image.....	44
5.2 ImageCollection.....	45

5.3 Geometry	46
5.4 Feature.....	47
5.5 FeatureCollection.....	47
5.6 Reducer	48
5.7 Join	53
บทที่ 6 การทำงานกับข้อมูลภาพดาวเทียม.....	54
6.1 การค้นหาข้อมูลจากคลังข้อมูล GEE	54
6.2 การเรียกดู properties.....	55
6.3 การแสดงผลภาพบนแผนที่	57
6.4 การกรองข้อมูล (filter).....	62
6.5 ช่วงคลื่น (Bands)	65
6.6 การคำนวณทางคณิตศาสตร์	68
6.7 การตัดภาพตามขอบเขตพื้นที่สนใจ (Clipping).....	73
6.8 การนำเข้าข้อมูล (Importing Data).....	75
6.9 การส่งออกข้อมูล (Exporting Data).....	78
บทที่ 7 การสร้างแอปพลิเคชัน Google Earth Engine (GEE Apps).....	85
7.1 GEE Apps คืออะไร	85
7.2 การวางแผนและออกแบบ GEE App.....	85
7.3 ออกแบบส่วนติดต่อผู้ใช้	86
7.4 ตัวอย่างการพัฒนาส่วนติดต่อผู้ใช้ของ GEE	86
7.5 การเผยแพร่ GEE App	94
บทที่ 8 การจำแนกการใช้ประโยชน์ที่ดินและวิเคราะห์ความเปลี่ยนแปลง	98
8.1 กำหนดพื้นที่วิเคราะห์และโหลดข้อมูลผึ้งสอน	98
8.2 การสร้างภาพ Composite จาก Sentinel-2 ในสองช่วงเวลา.....	98
8.3 การเลือกแบบจำลองสำหรับและสุมตัวอย่างข้อมูลผึ้งสอน	99
8.4 การผึ้งสอนแบบจำลอง.....	99
8.5 การกำหนดพารามิเตอร์เพื่อการแสดงผล.....	100
8.6 การแสดงผลลัพธ์บนแผนที่	100
8.7 การประเมินความแม่นยำ.....	101
บทที่ 9 การติดตามสถานการณ์ภัยพิบัติ	103
9.1 การวิเคราะห์พื้นที่น้ำท่วม	103
9.2 การวิเคราะห์พื้นที่เสี่ยงภัยแล้ง	106
เอกสารอ้างอิง	111

บทที่ 1: รู้จักกับ Google Earth Engine (GEE)

1.1 Google Earth Engine คืออะไร

Google Earth Engine (ต่อไปจะเรียกว่า GEE) เป็นแพลตฟอร์มบนระบบคลาวด์ที่ออกแบบมาสำหรับ การวิเคราะห์ข้อมูล ภูมิสารสนเทศ (Geospatial Data) ขนาดใหญ่ โดยเฉพาะข้อมูลภาพถ่ายดาวเทียมและข้อมูล เชิงพื้นที่อื่นๆ GEE ช่วยให้นักวิทยาศาสตร์ นักวิจัย และนักพัฒนาสามารถเข้าถึงคลังข้อมูลดาวเทียมขนาดใหญ่ ที่อัปเดตอย่างต่อเนื่อง พร้อมด้วยเครื่องมือประมวลผลประสิทธิภาพสูง เพื่อนำไปใช้ ในการศึกษาการเปลี่ยนแปลงของพื้นผิวโลก การจัดการทรัพยากรธรรมชาติ การติดตามผลกระทบจากภัยพิบัติ และงานด้านสิ่งแวดล้อมอื่นๆ อีกมากมาย โดยไม่ต้องกังวลเรื่องข้อจำกัดของทรัพยากรคอมพิวเตอร์ส่วนบุคคล สามารถเข้าถึงที่ <https://earthengine.google.com/>



Meet Earth Engine

Google Earth Engine combines a multi-petabyte catalog of satellite imagery and geospatial datasets with planetary-scale analysis capabilities. Scientists, researchers, and developers use Earth Engine to detect changes, map trends, and quantify differences on the Earth's surface. Earth Engine is now available for commercial use, and remains free for academic and research use.

ภาพที่ 1 เว็บไซต์ของ GEE

1.2 ความเป็นมา

Google Earth Engine (ต่อไปจะเรียกว่า GEE) เป็นแพลตฟอร์มคลาวด์เชิงพื้นที่ที่ Google เปิดตัวครั้งแรก เมื่อวันที่ 2 ธันวาคม พ.ศ. 2553 โดย Rebecca Moore และทีม Google Earth Outreach เพื่อให้การประมวลผลภาพถ่ายดาวเทียมและข้อมูลภูมิสารสนเทศขนาดใหญ่เป็นไปอย่างรวดเร็วผ่านอินเทอร์เฟซออนไลน์ นักวิจัยสามารถเข้าถึงแคตตาล็อกภาพถ่าย Landsat, MODIS, Sentinel และชุดข้อมูลภูมิอากาศ

ย้อนหลังกว่า 25 ปี พร้อมใช้เครื่องมือวิเคราะห์เพื่อวัดการเปลี่ยนแปลงของพื้นผืนโลก เช่น การตรวจจับการตัดไม้ทำลายป่าและการติดตามคุณภาพแหล่งน้ำ (Google, nd)

หลังเปิดตัว GEE ได้รับการพัฒนาอย่างต่อเนื่องในรูปแบบชุดข้อมูลขนาดใหญ่เพียงไบต์ (มากกว่า 80 เพตะไบต์) พร้อม API สำหรับ JavaScript และ Python ที่ใช้งานง่าย นักพัฒนาสามารถเขียนโค้ดเพื่อประมวลผลภาพทั้งในอินเตอร์เฟซ Code Editor หรือเรียกใช้งานผ่าน REST API ภายใต้ระบบ Cloud ของ Google ซึ่งช่วยลดภาระการจัดการเซิร์ฟเวอร์และเร่งความเร็วในการวิเคราะห์ข้อมูลระดับโลก (Moore, 2010)

ในปี พ.ศ. 2560 มีการตีพิมพ์บทความสำคัญโดย Gorelick et al. ในวารสาร *Remote Sensing of Environment* ซึ่งสรุปสถานะปัจจุบัน ภาระการจัดการเริร์กฟล็อว และกรณีศึกษาการประยุกต์ใช้ GEE ในงานวิจัยด้านป่าไม้ ภัยแล้ง และสิ่งแวดล้อมอื่นๆ บทความนี้ถือเป็นจุดเปลี่ยนที่ทำให้ GEE ถูกนำไปใช้ในงานวิชาการมากขึ้นและเป็นมาตรฐานใหม่ของการวิเคราะห์ภาพถ่ายดาวเทียมในระดับโลก (Gorelick, 2017)

GEE ได้ถูกนำไปประยุกต์ใช้อย่างกว้างขวางในการแก้ไขปัญหาสิ่งแวดล้อมและสังคมที่สำคัญ เช่น

- การติดตามการตัดไม้ทำลายป่า (Deforestation Monitoring) วิเคราะห์ภาพถ่ายดาวเทียมย้อนหลังเพื่อดูการเปลี่ยนแปลงของพื้นที่ป่าไม้ และแจ้งเตือนเมื่อมีการบุกรุกพื้นที่ป่า
- การประเมินผลกระทบจากการเปลี่ยนแปลงสภาพภูมิอากาศ (Climate Change Impact Assessment) ศึกษาการเปลี่ยนแปลงของราตรน้ำแข็ง ระดับน้ำทะเล การขยายตัวของทะเลราย
- การจัดการภัยพิบัติ (Disaster Management) ประเมินความเสี่ยงจากน้ำท่วม ไฟป่า แผ่นดินไหว และวางแผนการรับมือและพื้นที่
- การจัดการทรัพยากรน้ำ (Water Resource Management) ติดตามปริมาณน้ำในอ่างเก็บน้ำ การเปลี่ยนแปลงของพื้นที่ชุมชน้ำ และการใช้น้ำเพื่อการเกษตร
- การทำแผนที่การใช้ประโยชน์ที่ดิน (Land Use/Land Cover Mapping) จำแนกประเภทการใช้ประโยชน์ที่ดินและการเปลี่ยนแปลงเมื่อเวลาผ่านไป

1.3 ข้อดีและจุดเด่นของ GEE

จุดเด่นสำคัญที่สุดคือ การประมวลผลบนระบบคลาวด์ (Cloud Computing) ของ Google ทำให้ผู้ใช้ไม่จำเป็นต้องดาวน์โหลดข้อมูลภาพถ่ายดาวเทียมขนาดใหญ่มาเก็บไว้ในเครื่องคอมพิวเตอร์ส่วนตัว และไม่ต้องลงทุนกับฮาร์ดแวร์ที่ประสิทธิภาพสูงเพื่อการประมวลผล ผู้ใช้เพียงแค่มีอินเทอร์เน็ต ก็สามารถเข้าถึงและวิเคราะห์ข้อมูลได้จากทุกที่

GEE มีคลังข้อมูลขนาดใหญ่ (Petabyte-scale Data Catalog) ที่เก็บรวบรวมชุดข้อมูลภาพถ่ายดาวเทียมและข้อมูลภูมิสารสนเทศอื่นๆ มากกว่า 80 เพตะไบต์ (Petabytes) และยังคงเพิ่มขึ้นเรื่อยๆ ซึ่งรวมถึงข้อมูลจากดาวเทียมที่สำคัญ เช่น Landsat (ตั้งแต่ปี 1972) Sentinel-1 Sentinel-2 MODIS รวมถึงข้อมูลภูมิอากาศ ข้อมูลภูมิประเทศ (DEM) และข้อมูลทางสังคมและเศรษฐกิจต่างๆ ข้อมูลเหล่านี้พร้อมใช้งานสำหรับการวิเคราะห์ได้ทันที

GEE ถูกออกแบบมาเพื่อรองรับการประมวลผลข้อมูลจำนวนมากมหาศาลพร้อมๆ กัน (Parallel Processing) โดยอัตโนมัติ หรือ การประมวลผลแบบขนาน (Parallel Processing) ทำให้การวิเคราะห์ที่ซับซ้อนซึ่งอาจใช้เวลาหลายวันหรือหลายสัปดาห์บนคอมพิวเตอร์ทั่วไป สามารถทำได้เร็วสิ้นภายในเวลาไม่กี่นาทีหรือไม่กี่ชั่วโมง

GEE มี Application Programming Interface (API) ให้เลือกใช้ 2 ภาษาหลักคือ JavaScript (สำหรับการใช้งานผ่าน GEE Code Editor แบบ Web-based IDE) และ Python (สำหรับการใช้งานผ่านไลบรารี ee ใน Python) ทำให้ผู้ใช้สามารถพัฒนาสคริปต์และโมเดลการวิเคราะห์ที่ซับซ้อนได้ตามความต้องการ

GEE เปิดให้ใช้งานได้ฟรีสำหรับวัตถุประสงค์ที่ไม่ใช่เชิงพาณิชย์ เช่น งานวิจัย การศึกษา และโครงการเพื่อสาธารณะประโยชน์ ซึ่งเป็นการเปิดโอกาสให้นักวิจัย นักศึกษา และองค์กรพัฒนาเอกชนทั่วโลกสามารถเข้าถึงเทคโนโลยีนี้ได้ (สำหรับการใช้งานเชิงพาณิชย์ อาจจะมีเงื่อนไขและค่าใช้จ่ายตามที่ Google กำหนด)

1.4 ความสำคัญของ GEE ในการบริหารงานท้องถิ่น (Importance of GEE in Local Administration)

ความท้าทายขององค์กรปกครองส่วนท้องถิ่นในการเข้าถึงและใช้ข้อมูลภูมิสารสนเทศ ประกอบด้วย

- 1) ข้อจำกัดด้านงบประมาณในการจัดซื้อข้อมูลภาพถ่ายดาวเทียมหรือซอฟต์แวร์ราคาแพง
- 2) ข้อจำกัดด้านบุคลากรที่มีความเชี่ยวชาญเฉพาะทาง
- 3) ข้อจำกัดด้านโครงสร้างพื้นฐานคอมพิวเตอร์สมรรถนะสูงสำหรับการประมวลผลข้อมูลขนาดใหญ่
- 4) ความต้องการข้อมูลที่ทันสมัยและต่อเนื่องเพื่อการติดตามสถานการณ์

GEE ตอบโจทย์ความท้าทายเหล่านี้อย่างไร

- การเข้าถึงข้อมูลมหาศาลฟรี เพราะ GEE มีคลังข้อมูลภาพถ่ายดาวเทียม (Landsat, Sentinel, MODIS ฯลฯ) ข้อมูลภูมิอากาศ ภูมิประเทศ และข้อมูลเชิงพื้นที่อื่นๆ ที่หลากหลายและครอบคลุมทั่วโลก ย้อนหลังหลายสิบปี และมีการปรับปรุงให้ทันสมัยอยู่เสมอ
- แพลตฟอร์มประมวลผลบนคลาวด์ ไม่จำเป็นต้องใช้คอมพิวเตอร์สมรรถนะสูงหรือติดตั้งซอฟต์แวร์ที่ซับซ้อน ลดภาระค่าใช้จ่ายด้านハードแวร์และซอฟต์แวร์
- เครื่องมือวิเคราะห์ที่มีประสิทธิภาพ รองรับการวิเคราะห์ข้อมูลภูมิสารสนเทศขนาดใหญ่ ด้วย

อัลกอริทึมและฟังก์ชันที่หลากหลาย

ตัวอย่างการประยุกต์ใช้ GEE ในงานบริหารงานท้องถิน

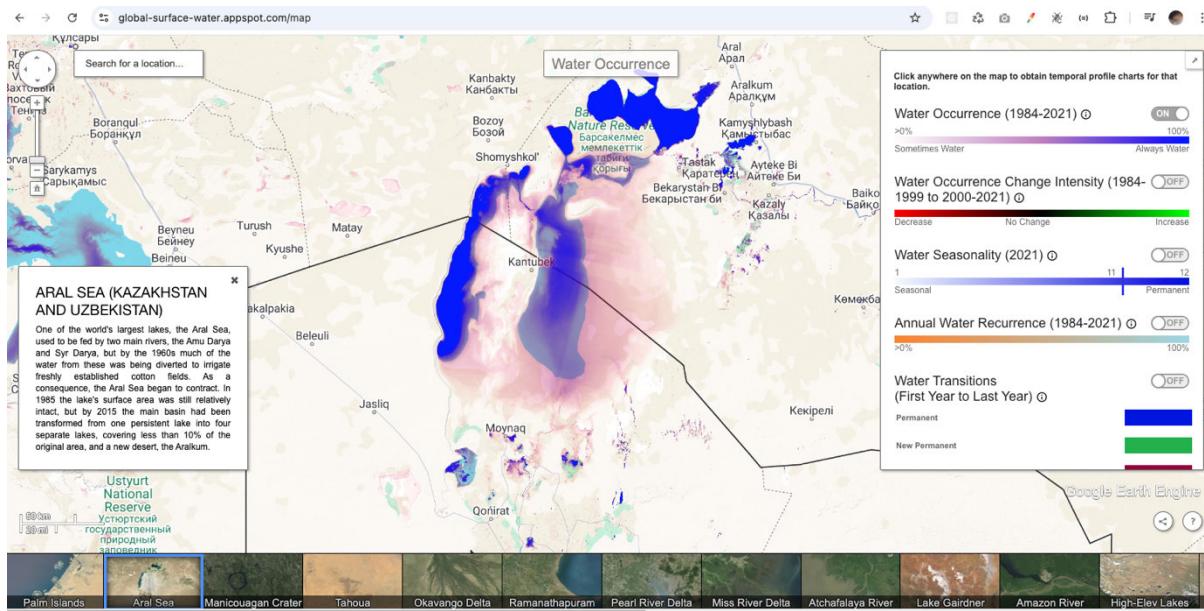
- ด้านการวางแผนเมืองและการติดตามการใช้ประโยชน์ที่ดิน ดูการเปลี่ยนแปลงพื้นที่สีเขียว การขยายตัวของเมือง การตรวจสอบการบุกรุกพื้นที่สาธารณะ
- ด้านการจัดการภัยพิบัติ การประเมินพื้นที่น้ำท่วม การติดตามพื้นที่เสี่ยงดินถล่ม การประเมินความเสียหายจากภัยแล้งหรือไฟป่า
- ด้านการจัดการทรัพยากรธรรมชาติและสิ่งแวดล้อม เช่น การติดตามการเปลี่ยนแปลงพื้นที่ป่าไม้ แหล่งน้ำ การประเมินคุณภาพน้ำเบื้องต้น (เช่น ความขุ่น) การติดตามผลกระทบจากการเกษตร
- ด้านการเกษตร เช่น การประเมินสุขภาพพืชพรรณ การคาดการณ์ผลผลิต การติดตามภัยแล้งในพื้นที่เกษตร
- ด้านการพัฒนาโครงสร้างพื้นฐาน เช่น การติดตามการเปลี่ยนแปลงของเส้นทางคมนาคม การประเมินผลกระทบจากการก่อสร้าง

ประโยชน์ที่ท้องถินจะได้รับ

การตัดสินใจบนฐานข้อมูลที่ถูกต้องแม่นยำ (Evidence-based decision making) เพิ่มประสิทธิภาพการทำงานลดตนทุน และเพิ่มความโปร่งใส

ตัวอย่างการนำไปใช้งาน

โครงการ: JRC Global Surface Water Explorer (GSWE) (European Commission, 2020) JRC ใช้ GEE ในการประมวลผลภาพถ่ายดาวเทียม Landsat ทั้งหมด (มากกว่า 4 ล้านภาพ) ตั้งแต่ปี 1984 เพื่อสร้างชุดข้อมูลความละเอียด 30 เมตร ที่แสดงการกระจายตัวและการเปลี่ยนแปลงของแหล่งน้ำผิวดินทั่วโลกเป็นรายเดือน



ภาพที่ 2 โครงการ JRC Global Surface Water Explorer (GSWE)

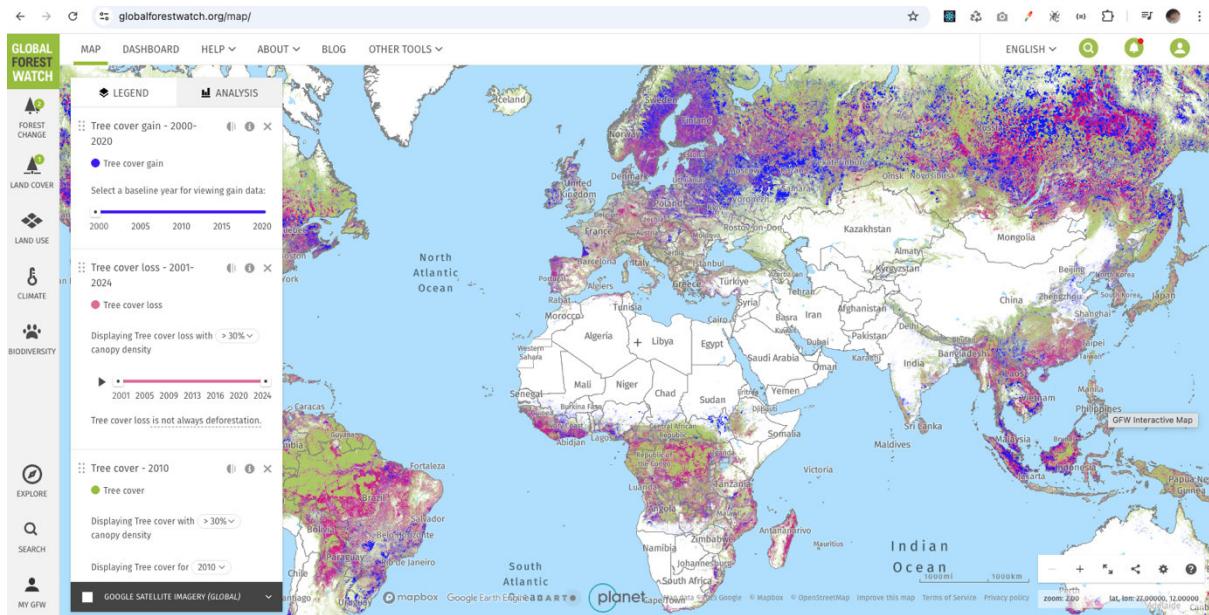
ที่มา <https://global-surface-water.appspot.com/map>

โครงการ: การติดตามการตัดไม้ทำลายป่าและการเปลี่ยนแปลงพื้นที่ป่า (Forest Monitoring and Deforestation Alerts) โดย Joint Research Centre (JRC) ของคณะกรรมการยุโรป ประกอบด้วย University of Maryland (GLAD Lab), World Resources Institute (Global Forest Watch), FAO (SEPAL Platform) (Global Forest Watch, n.d.)

GLAD Alerts: ใช้ข้อมูล Landsat ใน GEE เพื่อสร้างระบบแจ้งเตือนการสูญเสียพื้นที่ป่าในรายสัปดาห์ ความละเอียด 30 เมตร <https://glad.earthengine.app/>

Global Forest Watch: เป็นแพลตฟอร์มที่ใช้ GEE (และเทคโนโลยีอื่น ๆ) ในการรวบรวม และแสดงผล และวิเคราะห์ข้อมูลพื้นที่ป่าไม้ทั่วโลก รวมถึง GLAD Alerts <https://www.globalforestwatch.org/map/>

FAO SEPAL (System for Earth Observation Data Access, Processing and Analysis for Land Monitoring): เป็นแพลตฟอร์มบนคลาวด์ที่ใช้งานง่าย ช่วยให้ประเทศต่าง ๆ เข้าถึงและประเมินผลข้อมูลดาวเทียมผ่าน GEE เพื่อการติดตามป่าไม้และการรายงาน (เช่น REDD+)

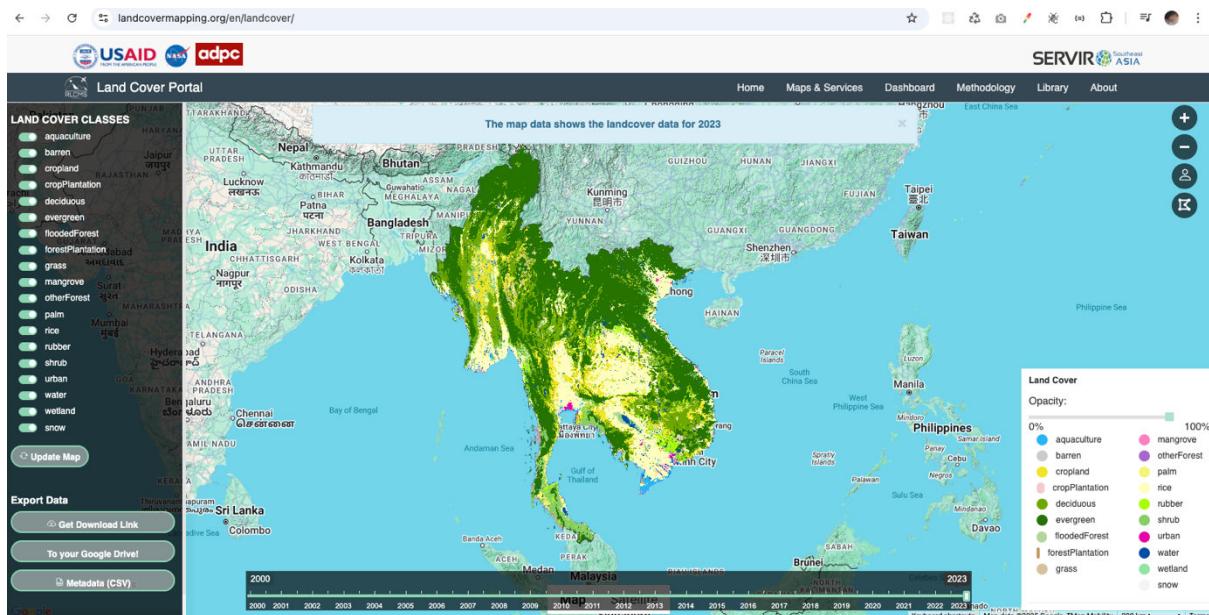


ภาพที่ 3 โครงการการติดตามการตัดไม้ทำลายป่าและการเปลี่ยนแปลงพื้นที่ป่า

ที่มา <https://www.globalforestwatch.org/map/>

โครงการ: SERVIR (NASA & USAID Initiative) โดย NASA, USAID และหน่วยงานพันธมิตรในภูมิภาคต่างๆ

<https://landcovermapping.org/en/landcover/> (SERVIR-SEA., nd)



ภาพที่ 4 โครงการ SERVIR (NASA & USAID Initiative)

ที่มา <https://landcovermapping.org/en/landcover/>

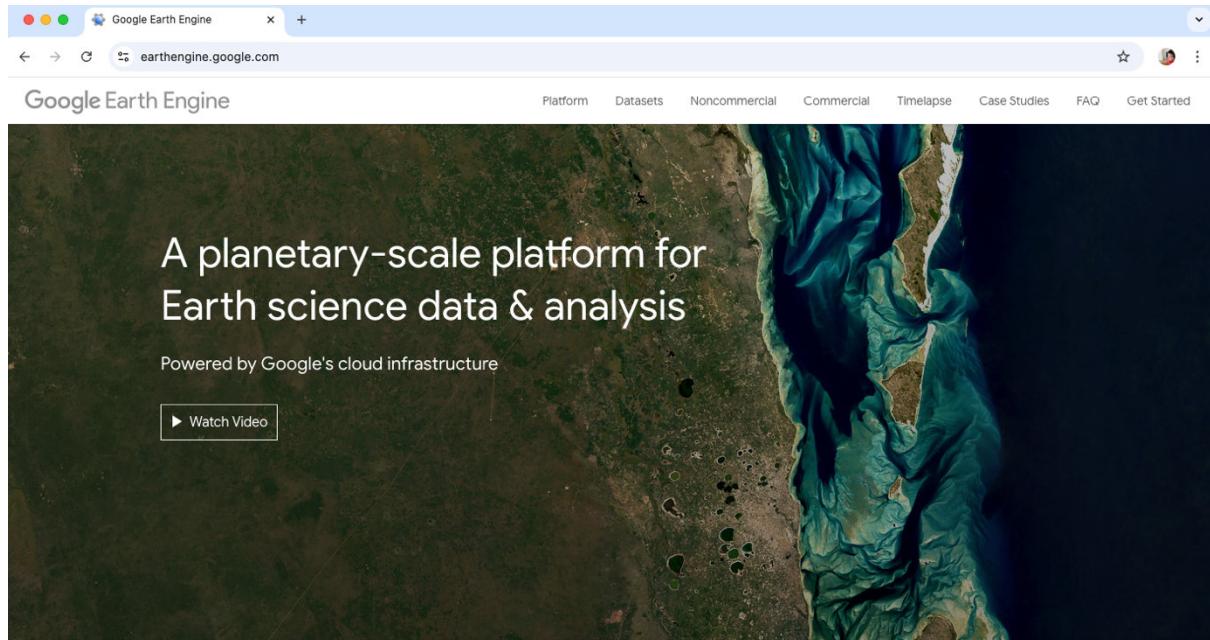
บทที่ 2: การใช้งาน GEE Code Editor

2.1 การสมัครใช้งานและการเข้าถึง GEE Code Editor

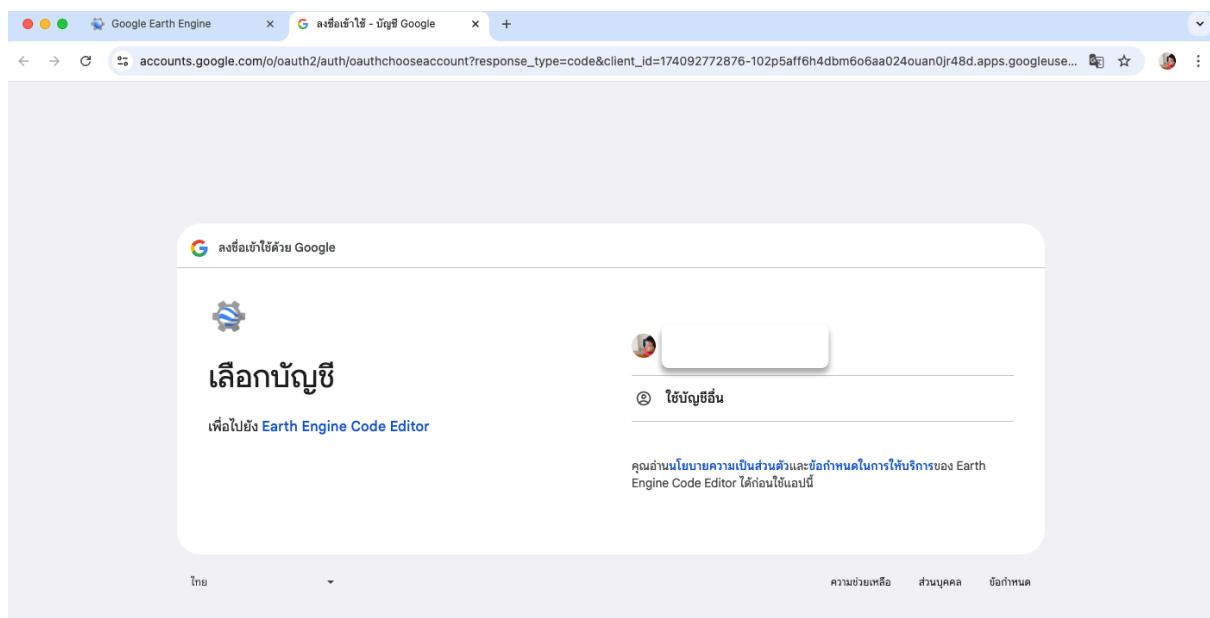
การเริ่มต้นใช้งาน Google Earth Engine นั้นไม่ซับซ้อน โดยผู้ที่สนใจสามารถสมัครเข้าใช้งานได้โดยไม่มีค่าใช้จ่ายสำหรับวัตถุประสงค์ส่วนบุคคล การศึกษา และงานวิจัยที่ไม่ใช่เชิงพาณิชย์

2.1.1 ขั้นตอนการสมัครบัญชี GEE

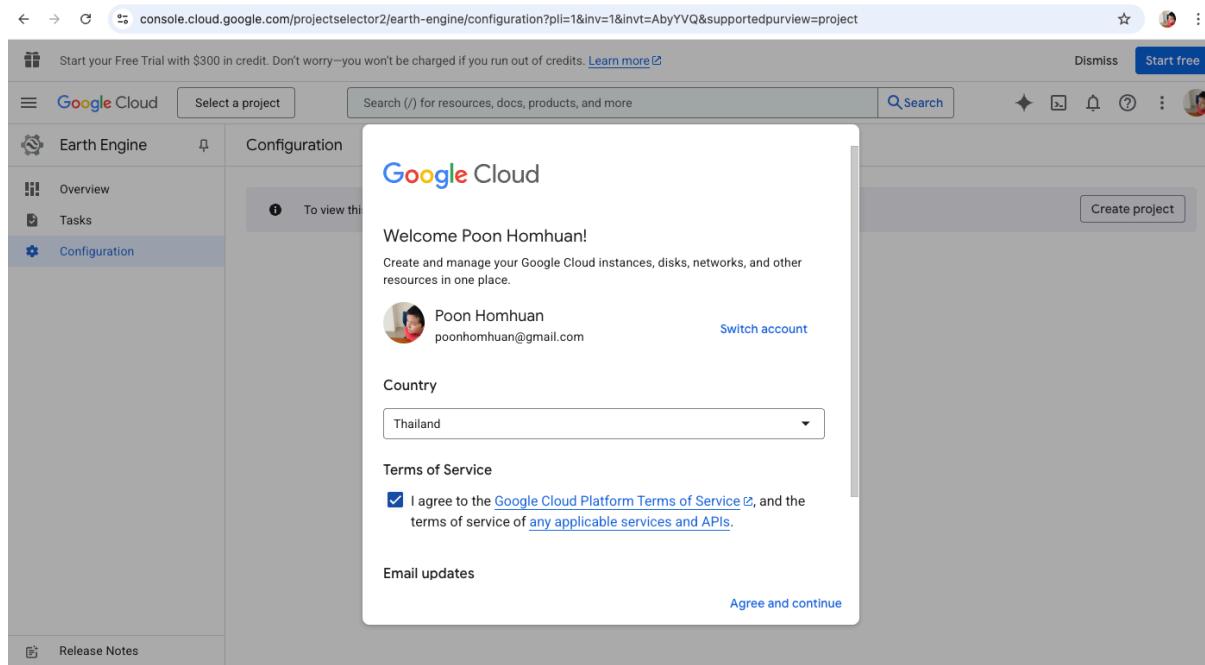
- 1) เข้าสู่เว็บไซต์ GEE โดยเปิดเบราว์เซอร์แล้วไปที่ URL: <https://earthengine.google.com>



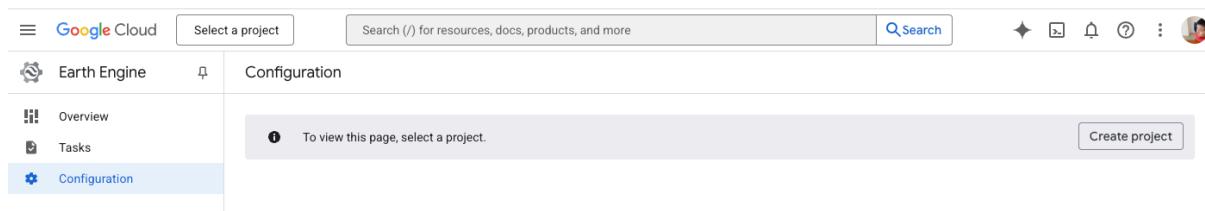
- 2) คลิก "Sign Up" หรือ "Get Started" บนหน้าเว็บไซต์
- 3) GEE จะใช้บัญชี Google (เช่น Gmail) ในการยืนยันตัวตนและการเข้าใช้งาน หากยังไม่มีบัญชี Google จำเป็นต้องสมัครก่อน



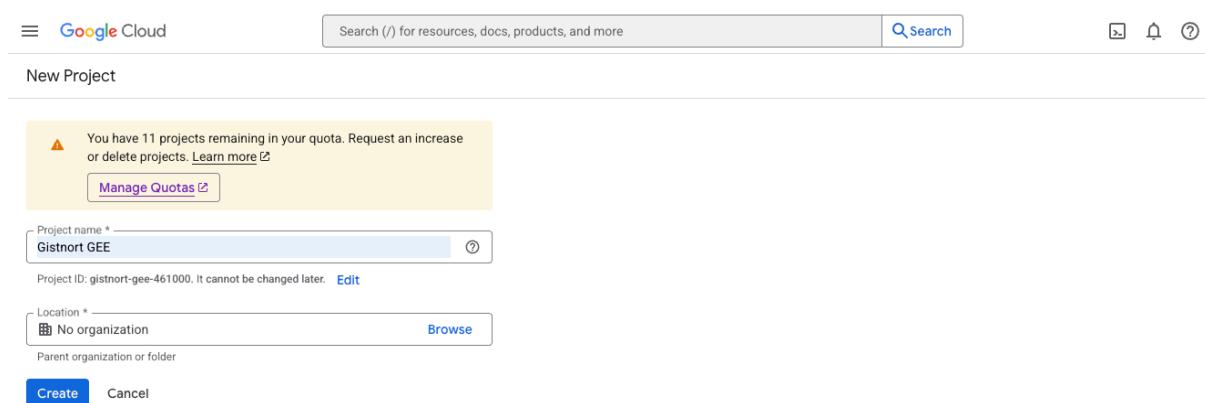
4) ระบบจะนำไปยังหน้า Google Cloud เพื่อเปิดใช้งาน Cloud ของ Google ขั้นตอนนี้เลือกยอมรับการใช้งาน Google Cloud



5) เลือก Create project



6) ใส่ชื่อของ Project > จากนั้น คลิก Create



7) เลือกลงทะเบียนแบบ noncommercial use

The screenshot shows the Google Cloud Earth Engine Configuration interface. The 'Configuration' tab is active. It provides instructions to register your Cloud project as either 'commercial' or 'noncommercial'. It highlights that Earth Engine is a powerful tool for geospatial analysis used for climate, sustainability, and environmental work. Unpaid access is offered to eligible noncommercial organizations, while businesses and governments engaging in operational activities access Earth Engine via paid commercial use. Two main sections are shown: 'Register for commercial use' (with a 'Continue' button) and 'See if you are eligible for noncommercial use' (with a 'Get started' button).

8) เลือก organization type แบบ Other

The screenshot shows the 'Register' page under the 'Configuration' section. Step 1, 'Select your organization type', is active. It asks 'Which of the following best describes you or your organization? *' and provides a dropdown menu with 'Other' selected. A 'Next' button is visible. To the right, a vertical list of steps is shown: 2. Check noncommercial eligibility, 3. Choose your plan, 4. Describe your work, and 5. Review summary. At the bottom are 'Register' and 'Cancel' buttons.

9) จักนั้นเลือก Individual research or noncommercial use และกรอกวัตถุประสงค์ในการใช้งาน GEE (Project description/motivation)

ในส่วนนี้เราควรอธิบายให้ชัดเจนว่าต้องการนำ GEE ไปใช้ประโยชน์ด้านใด เช่น "เพื่อศึกษาวิจัยการเปลี่ยนแปลงพื้นที่ป่าไม้ในภาคเหนือของประเทศไทย" หรือ "เพื่อใช้ในการเรียนการสอนวิชาการรับรู้จากระยะไกล" การให้ข้อมูลที่ชัดเจนจะช่วยให้การอนุมัติเป็นไปได้รวดเร็วขึ้น

Google Cloud Gistnort GEE Search (/) for resources, docs, products, and more Search

Earth Engine / Configuration / Register

Overview Tasks Configuration

Register

Select your organization type

Check noncommercial eligibility

Which of the following best describes your use case? *

Individual research or noncommercial use (not involved with an org...)

Please provide more info about how you will be using Earth Engine *

For use in remote sensing studies with Google Earth Engine

Based on your answers, you are eligible for noncommercial Earth Engine use.

Next

Choose your plan

Describe your work

Review summary

Release Notes

Register Cancel

10) คลิก Next

Google Cloud Gistnort GEE Search (/) for resources, docs, products, and more Search

Earth Engine / Configuration / Register

Overview Tasks Configuration

Register

Select your organization type

Check noncommercial eligibility

Choose your plan

A pricing plan is not required for noncommercial registration.

Next

Describe your work

Review summary

Register Cancel

11) เลือกรายละเอียดของเราว่าจะใช้ใน GEE

Google Cloud Gistnort GEE Search (/) for resources, docs, products, and more

Earth Engine / Configuration / Register

Overview Tasks Configuration

Register

- 1 Select your organization type
- 2 Check noncommercial eligibility
- 3 Choose your plan
- 4 Describe your work

Does your work with Earth Engine fall into any of these categories?

Mitigation
e.g., reduction or avoidance of greenhouse gas emissions / CO₂ equivalent

Adaptation
e.g., helping people and communities adapt to the impacts of climate change

Protection & conservation
e.g., land and ocean-based interventions to conserve biodiversity and ecosystems

Will you use Earth Engine for any of the following? *

Agriculture, Air Quality / Pollution, and Artificial intelligence and machine learning (AI / ML)

Next

12) คลิก Register

Google Cloud Gistnort GEE Search (/) for resources, docs, products, and more

Earth Engine / Configuration / Register

Overview Tasks Configuration

Register

Please provide more info about how you will be using Earth Engine

For use in remote sensing studies with Google Earth Engine

Your work

Does your work with Earth Engine fall into any of these categories?
Adaptation, Protection & conservation

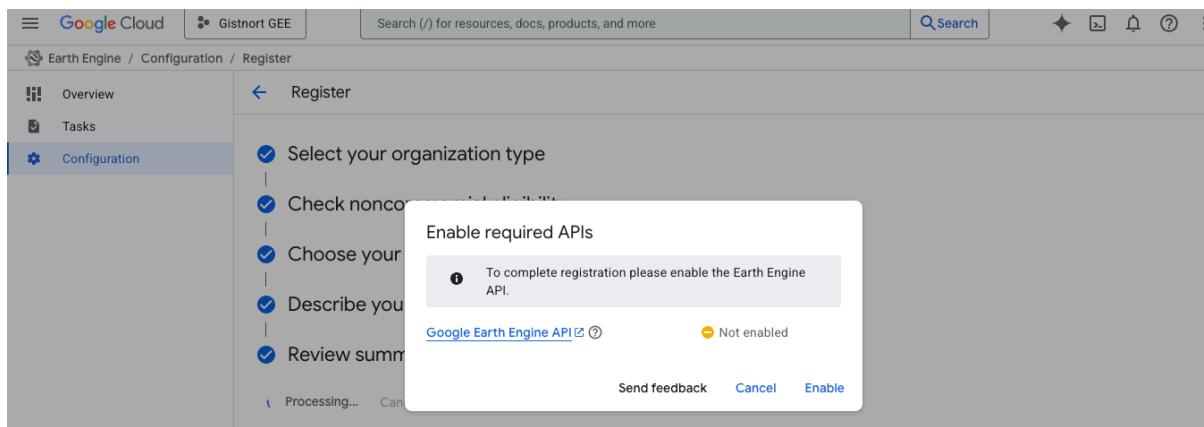
Will you use Earth Engine for any of the following?
Agriculture, Air Quality / Pollution, Artificial intelligence and machine learning (AI / ML)

To make changes, click the relevant step title and edit your answers.

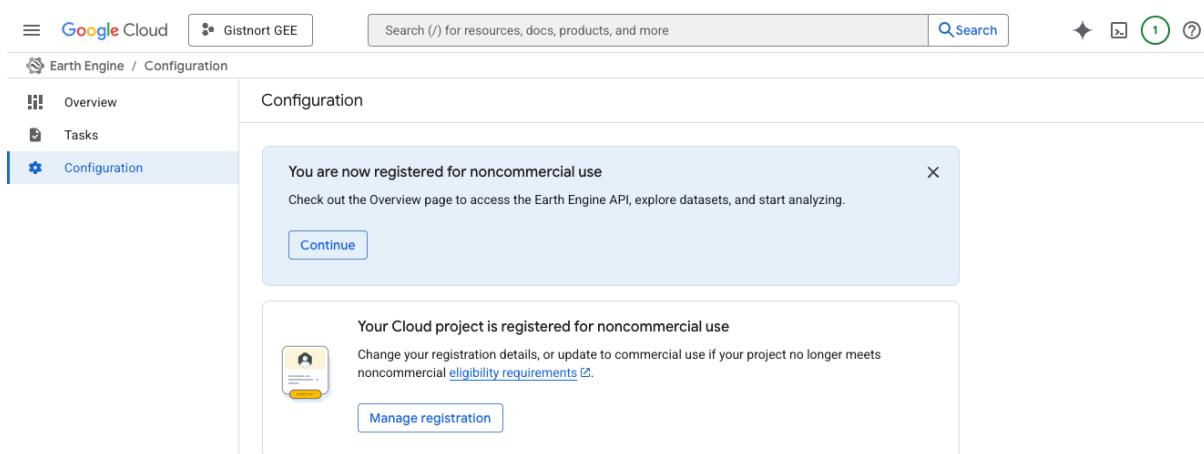
This information is collected to verify noncommercial eligibility, inform product improvements, and assess the sustainability impact of Earth Engine usage, subject to the [Google Cloud Privacy Notice](#).

Register Cancel

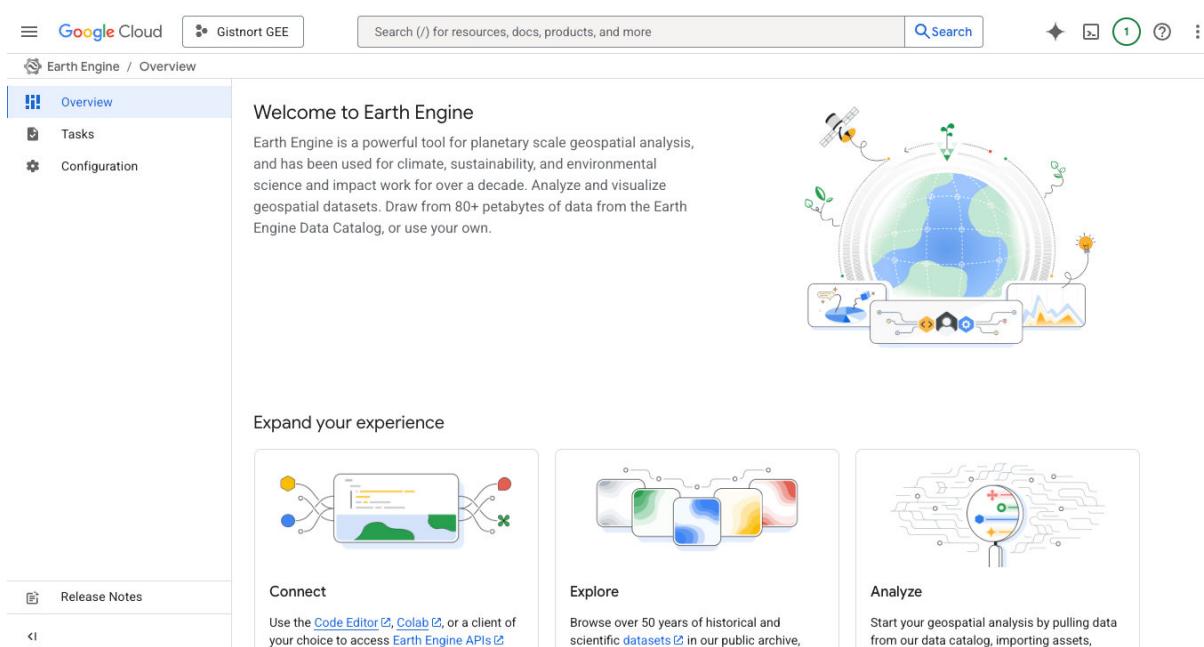
13) เลือก Enable GEE API



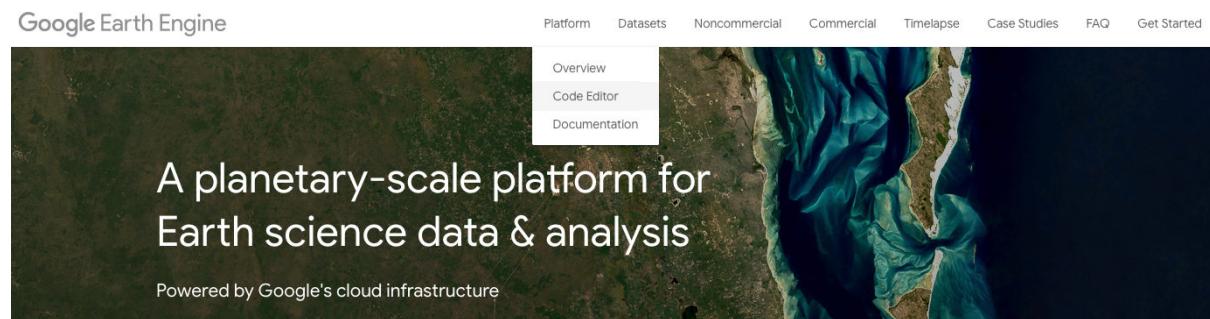
14) คลิก Continue เพื่อเข้าไปยัง GEE Cloud API



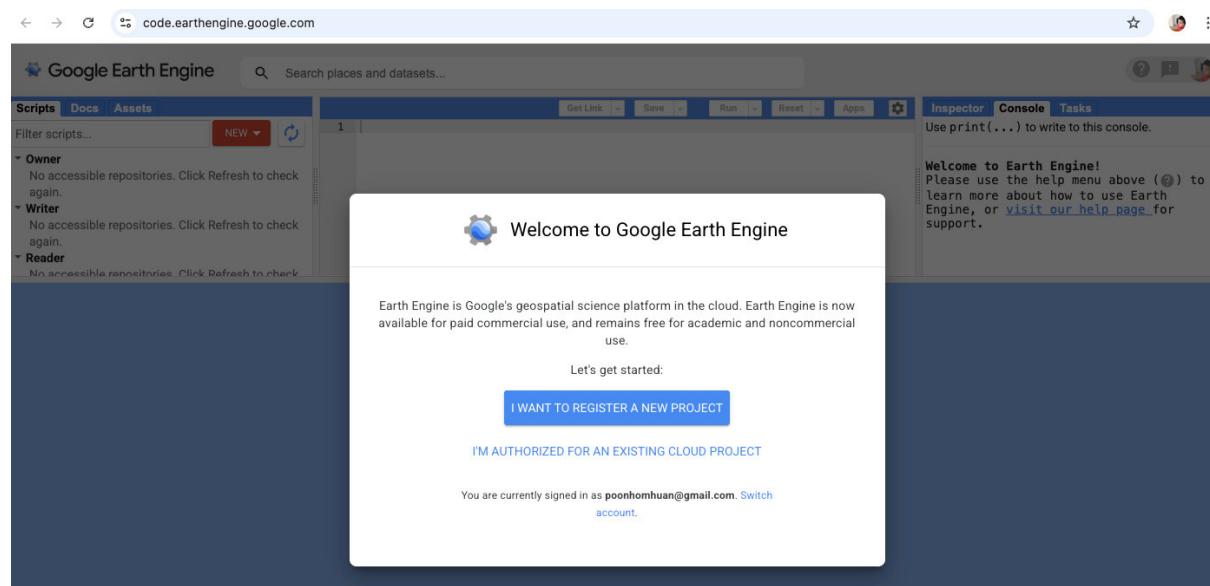
15) มองหา Card ชื่อ Connect จากนั้น คลิกที่ Code Editor



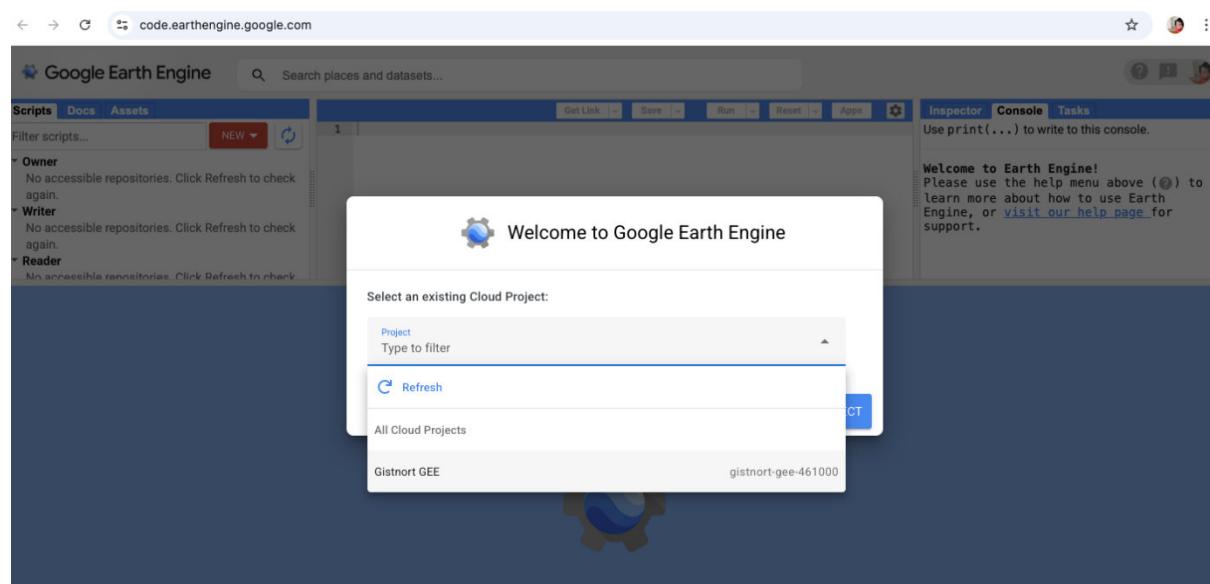
หรือ ก็ลับมายัง <https://earthengine.google.com> และ คลิกที่ Platform > เลือก Code Editor



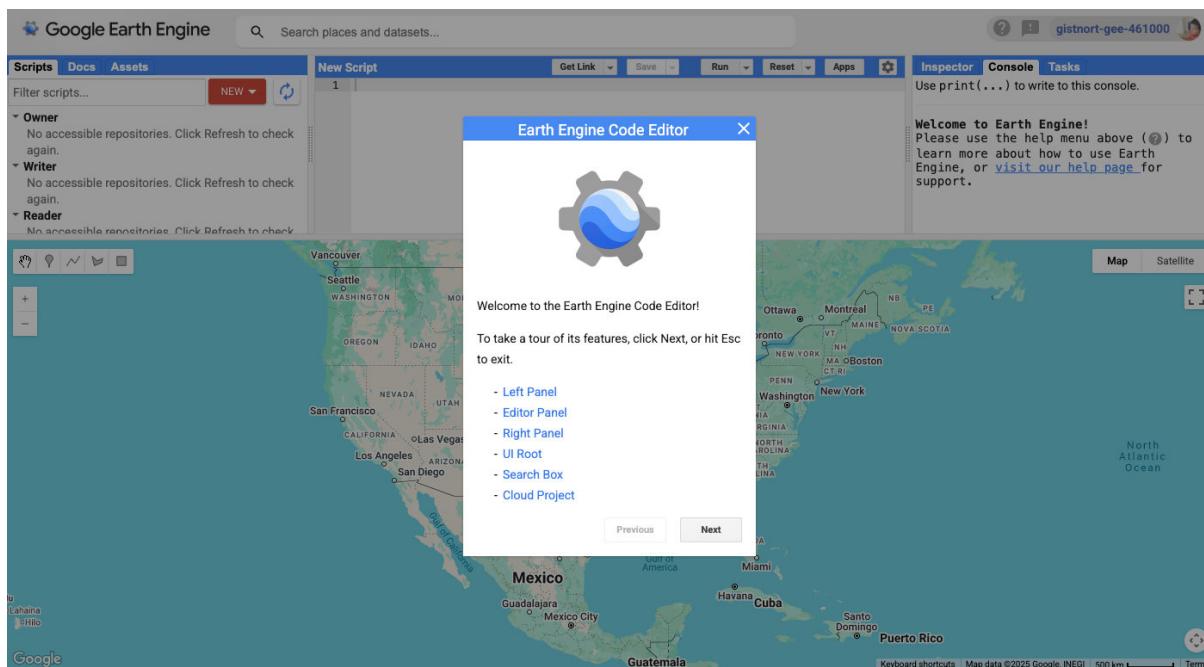
16) จากนั้นเลือก I'm Authorized for an existing cloud project



17) เลือก cloud project ที่เราสร้างขึ้น

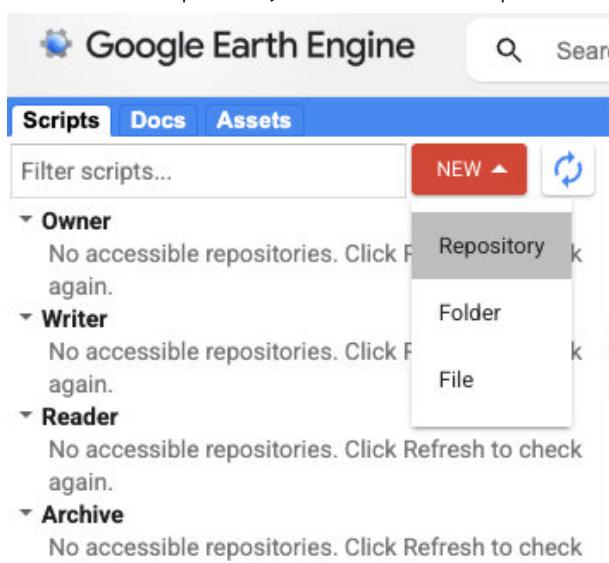


18) ระบบจะนำมายังหน้าของ GEE code editor

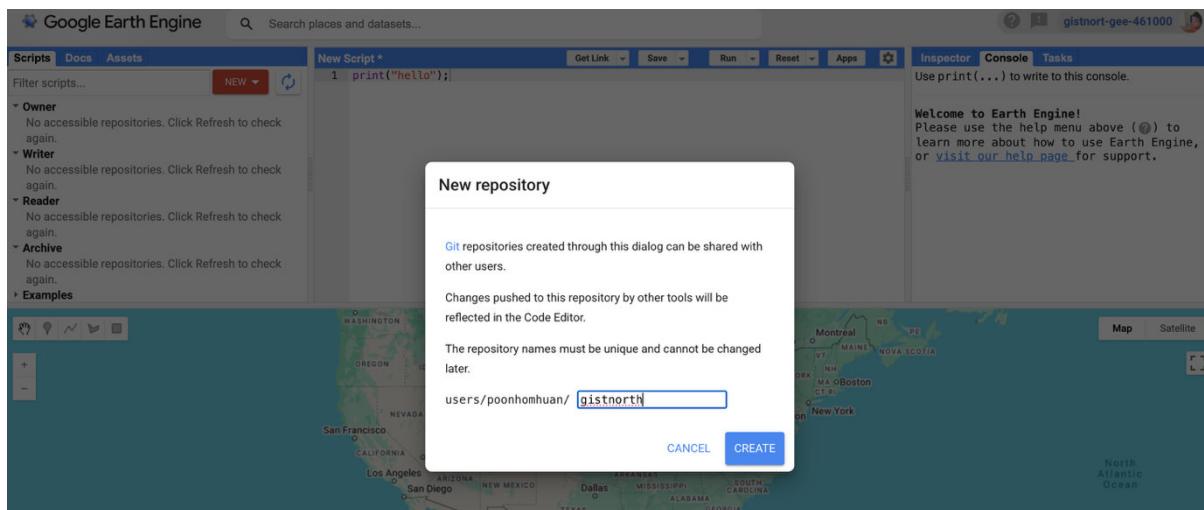


2.1.2 การเข้าสู่ระบบและการใช้งาน GEE Code Editor เป็นต้น

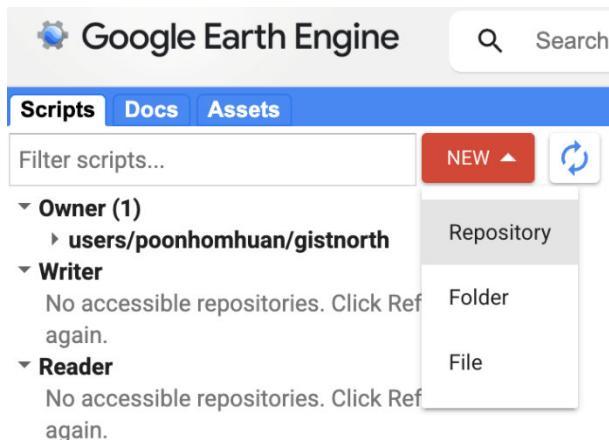
- 1) หลังจากได้รับการอนุมัติแล้ว สามารถเข้าใช้งาน GEE Code Editor ได้โดยตรงผ่านทาง URL <https://code.earthengine.google.com> และลงชื่อเข้าใช้ด้วยบัญชี Google ที่ได้รับการอนุมัติ
- 2) เมื่อเข้าสู่ระบบ จะพบกับหน้าจอหลักของ Code Editor ซึ่งเป็นหน้าต่างสำหรับการเขียนโค้ด JavaScript เพื่อสั่งงาน GEE โดยมีส่วนประกอบหลักๆ ที่จะอธิบายในบทถัดไป (เช่น Scripts, Docs, Assets, Console, Inspector, และแผนที่แสดงผล)
- 3) การสร้าง Repository สำหรับจัดเก็บ script > โดยไปที่ Repository



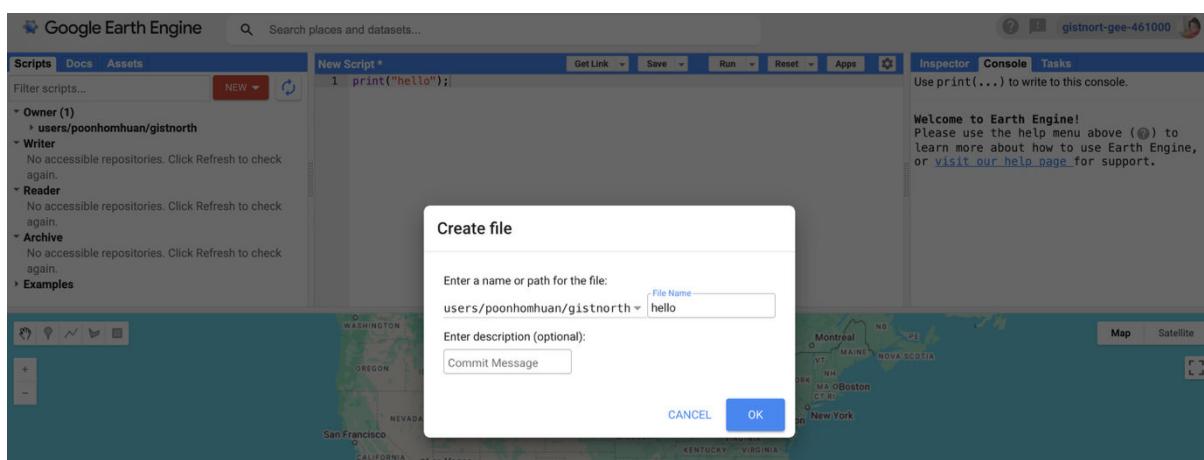
จากนั้นกำหนดชื่อ Repository



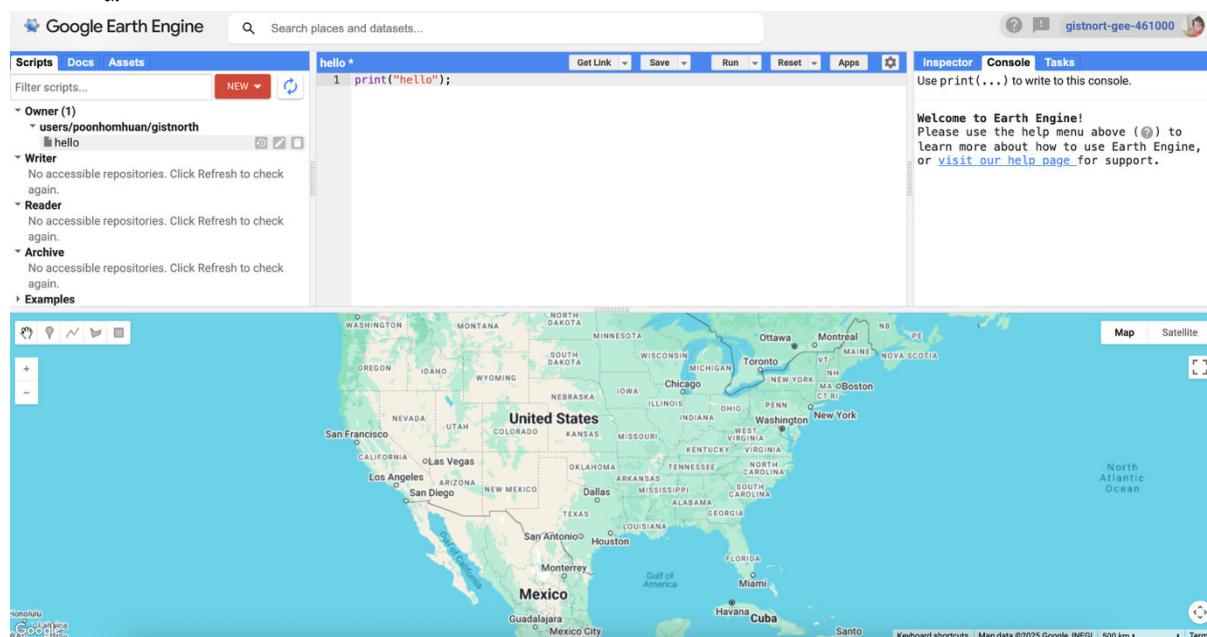
4) สร้างสคริปต์ใหม่ (New Script) สามารถสร้างไฟล์สคริปต์ใหม่ได้จากเมนู "File" > "New" > File



ใช้ชื่อ File > Ok



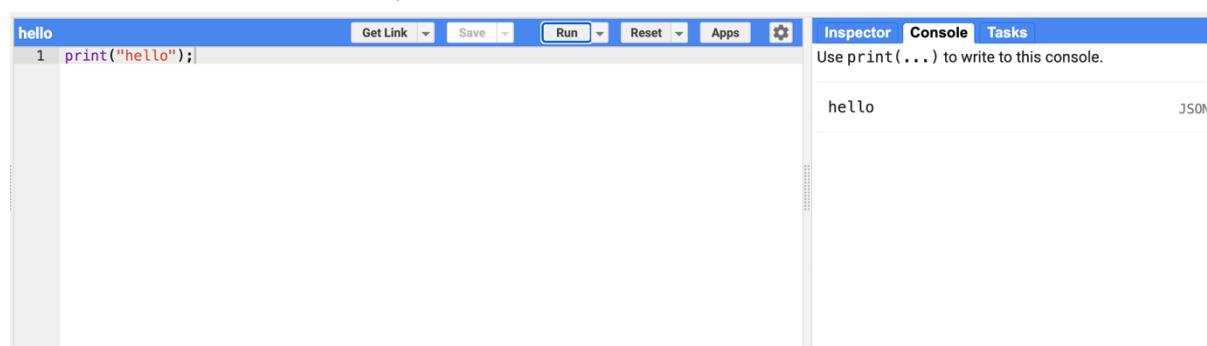
จะประภากูไฟล์ภายในไฟล์ Repository ที่สร้างขึ้น



เมื่อเขียนโค้ดเสร็จสิ้นหรือต้องการบันทึกสคริปต์ (Save Script) ให้คลิกที่ปุ่ม "Save" หรือ "Save as..." เพื่อตั้งชื่อและบันทึกสคริปต์ สคริปต์จะถูกเก็บไว้ใน Repository ส่วนตัวของผู้ใช้

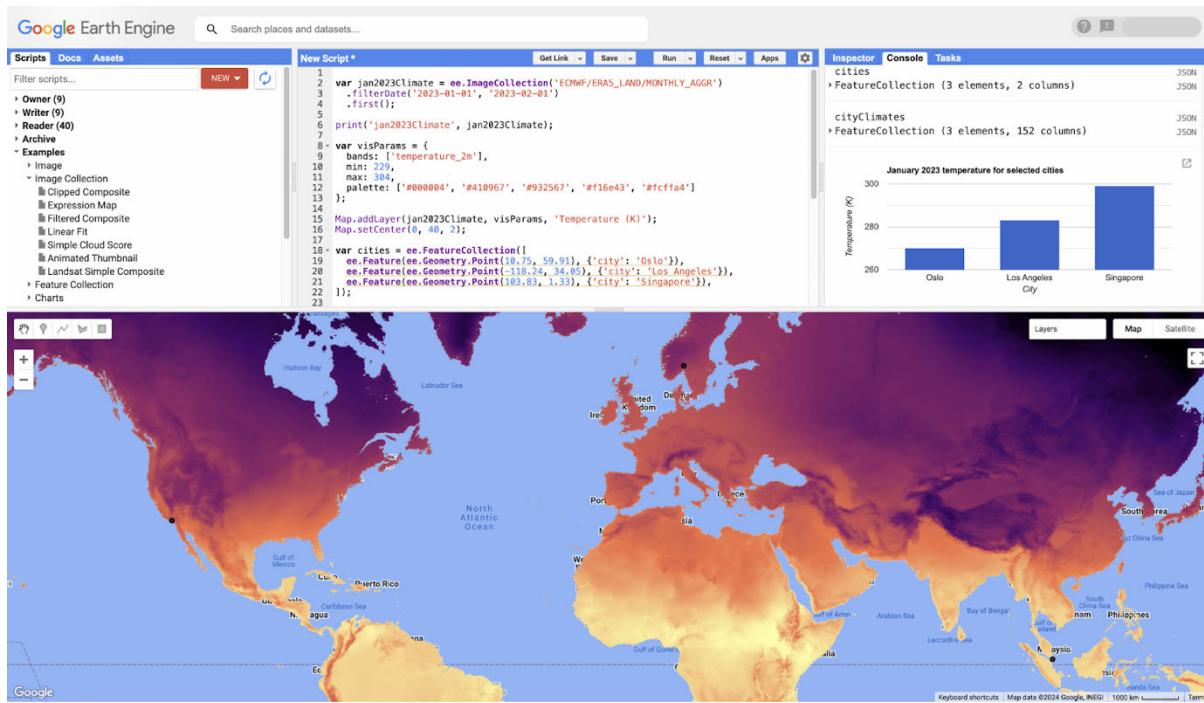
การเปิดสคริปต์เก่า (Open Script) สามารถเปิดสคริปต์ที่เคยบันทึกไว้ได้จาก "Scripts" โดยเลือกจาก Repository ของตนเอง (Owner) สคริปต์ที่ผู้อื่นแชร์มา (Writer, Reader) หรือตัวอย่างสคริปต์ (Examples)

5) หลังจากเขียนโค้ดใน Editor และ สามารถสั่งให้สคริปต์ทำงานได้โดยคลิกที่ปุ่ม "Run" ผลลัพธ์จากการทำงาน เช่น การแสดงภาพบนแผนที่ หรือการพิมพ์ข้อมูลออกมา จะประภากูในส่วน "Console" หรือบนแผนที่โดยตรง การทดลองรันสคริปต์ง่ายๆ



2.2 ส่วนประกอบต่างๆ ของ GEE Code Editor

GEE Code Editor ประกอบด้วยส่วนการทำงานหลักๆ ที่ช่วยอำนวยความสะดวกในการเขียน จัดการ และตรวจสอบสคริปต์ ดังนี้



2.2.1 Scripts

Scripts เป็นส่วนจัดการสคริปต์ โดยทั่วไปจะอยู่ทางด้านซ้ายบนของหน้าจอเป็นพื้นที่สำหรับจัดการไฟล์สคริปต์ JavaScript (.js) ของผู้ใช้ แบ่งออกเป็นส่วนย่อย ๆ ดังนี้

- 1) Owner สคริปต์ที่ผู้ใช้สร้างและเป็นเจ้าของ จะถูกจัดเก็บไว้ใน Repository ส่วนตัวบน Google Cloud ผู้ใช้สามารถสร้างโฟลเดอร์เพื่อจัดระเบียบสคริปต์ได้
- 2) Writer/Reader สคริปต์ที่ผู้ใช้อ่านและรีวิวได้ โดยอาจมีสิทธิ์ในการแก้ไขหรืออ่านอย่างเดียว
- 3) Examples ชุดตัวอย่างสคริปต์ที่ทาง GEE เตรียมไว้ให้ เพื่อเป็นแนวทางในการเรียนรู้ฟังก์ชันและการใช้งานต่างๆ
- 4) Archive สคริปต์ที่ถูกลบไปแล้ว (สามารถคืนได้ภายในระยะเวลาหนึ่ง) ในส่วนนี้ผู้ใช้สามารถสร้างสคริปต์ใหม่ เปิดสคริปต์ที่เมื่อยุบ บันทึก เปลี่ยนชื่อ ลบ หรือแชร์สคริปต์ให้กับผู้อื่นได้จากส่วนนี้

Google Earth Engine

Search |

Scripts Docs Assets

Filter scripts... NEW ▾

- ▼ Owner (1)
 - ▶ users/sakdahomhuan/gg_engine
- ▼ Writer

No accessible repositories. Click Refresh to check again.
- ▶ Reader (3)
 - ▶ Archive
 - ▶ Examples
 - ▶ gg_engine

2.2.2 Docs

Docs เป็นส่วนเอกสารอ้างอิง ถัดมาจากการ Scripts มีหน้าที่เป็นแหล่งรวมเอกสารอ้างอิง (API Documentation) สำหรับฟังก์ชัน อัลกอริทึม และชุดข้อมูลทั้งหมดที่มีใน GEE ผู้ใช้สามารถค้นหาข้อมูลเกี่ยวกับวิธีการใช้งานพารามิเตอร์ต่างๆ ของฟังก์ชัน ประเภทข้อมูลที่รับเข้าและส่งออก และตัวอย่างโค้ดสั้นๆ ในส่วนนี้ถูกใช้งานเมื่อต้องการทราบรายละเอียดของฟังก์ชันใดๆ เช่น ee.Image() หรือ Map.addLayer() สามารถพิมพ์ชื่อฟังก์ชันในช่องค้นหา หรือเลือดูตามหมวดหมู่ได้ ข้อมูลในส่วน Docs มีประโยชน์อย่างยิ่งในระหว่างการเขียนโค้ด

Google Earth Engine

Search |

Scripts Docs Assets

Filter methods...

- ▶ ee.Algorithms
- ▶ ee.Array
- ▶ ee.Blob
- ▶ ee.Classifier
- ▶ ee.Clusterer
- ▶ ee.ConfusionMatrix
- ▶ ee.Date
- ▶ ee.DateRange
- ▶ ee.Dictionary
- ▶ ee.ErrorMargin
- ▶ ee.Feature
- ▶ ee.FeatureCollection
- ▶ ee.Filter
- ▶ ee.Geometry
- ▶ ee.Image

2.2.3 Assets

Assets เป็นส่วนจัดการข้อมูลส่วนตัว โดยทั่วไปจะอยู่ทางด้านซ้าย ถัดลงมาจากระดับ Docs เป็นพื้นที่สำหรับจัดเก็บและจัดการข้อมูลส่วนตัวของผู้ใช้ (Private Assets) ที่อัปโหลดเข้าไปใน GEE เช่น ข้อมูล Shapefile, GeoTIFF, CSV หรือ Table ที่ได้จากการ Export งานใน GEE ข้อมูลที่เก็บใน Assets สามารถนำมาเรียกใช้ในสคริปต์โดยตรง ในส่วนนี้เราสามารถอัปโหลดข้อมูลใหม่ สร้าง Image Collection หรือ Feature Collection จากข้อมูลที่อัปโหลด จัดการสิทธิ์การเข้าถึงข้อมูล และดูรายละเอียดของ Asset แต่ละรายการได้จากส่วนนี้

2.2.4 Console

Console เป็นส่วนแสดงผลลัพธ์และข้อผิดพลาดตำแหน่ง โดยทั่วไปจะอยู่ทางด้านขวาของหน้าจอ แบ่งเป็นแท็บอยู่ๆ หน้าที่เป็นส่วนที่ใช้แสดงผลลัพธ์จากการทำงานของสคริปต์ และแจ้งเตือนข้อผิดพลาด (Error messages) ที่เกิดขึ้น

```

New Script *
Get Link Save Run Reset Apps
Inspector Console Tasks
Use print(...) to write to this console.

1 var geometry = ee.Geometry.Polygon([
2   [[98.9171009716561, 18.815619476862654],
3   [98.9171009716561, 18.68557890893941],
4   [99.0873890575936, 18.68557890893941],
5   [99.0873890575936, 18.815619476862654]]]);
6 var collection = ee.ImageCollection('COPERNICUS/S2')
7   .filterDate('2021-01-01', '2021-01-31')
8   .filterBounds(geometry)
9   .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
10  .select(['B4', 'B3', 'B2'])
11  .median();
12 print(collection)
13 Map.centerObject(geometry, 10);
14 Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Median'

```

2.2.5 Inspector

Inspector เป็นส่วนตรวจสอบข้อมูลบนแผนที่ เป็นแท็บหนึ่งในส่วน Console (ด้านขวา) ทำหน้าที่ เป็นเครื่องมือที่ช่วยให้ผู้ใช้สามารถคลิกบนแผนที่เพื่อตรวจสอบค่า Pixel ของ Image Layers ที่แสดงอยู่ ณ ตำแหน่งนั้นๆ รวมถึงคุณสมบัติ (Properties) ของ Feature ที่ถูกกดหรือแสดงผลบนแผนที่

ด้านการใช้งาน เมื่อเปิดใช้งานแท็บ Inspector แล้วคลิกที่ชุดใดๆ บนแผนที่ จะปรากฏข้อมูลของ Layer ต่างๆ ณ จุดนั้น เช่น ค่า Band ของภาพถ่ายดาวเทียม ค่าความสูงจาก DEM หรือ Properties ของ Vector ที่เลือกข้อมูลนี้มีประโยชน์ในการสำรวจข้อมูลเบื้องต้น



```

New Script * Get Link Save Run Reset Apps
1 var geometry = ee.Geometry.Polygon(
2   [[[-98.9171009716561, 18.815619476862654], ->
3     [-98.9171009716561, 18.68557890893041], ->
4     [-99.0873890575936, 18.68557890893041], ->
5     [-99.0873890575936, 18.815619476862654]]]);
6 var collection = ee.ImageCollection('COPERNICUS/S2')
7 .filterDate('2021-01-01', '2021-01-31')
8 .filterBounds(geometry)
9 .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
10 .select(['B4', 'B3', 'B2'])
11 .median();
12 print(collection)
13 Map.centerObject(geometry, 10);
14 Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Median')

```

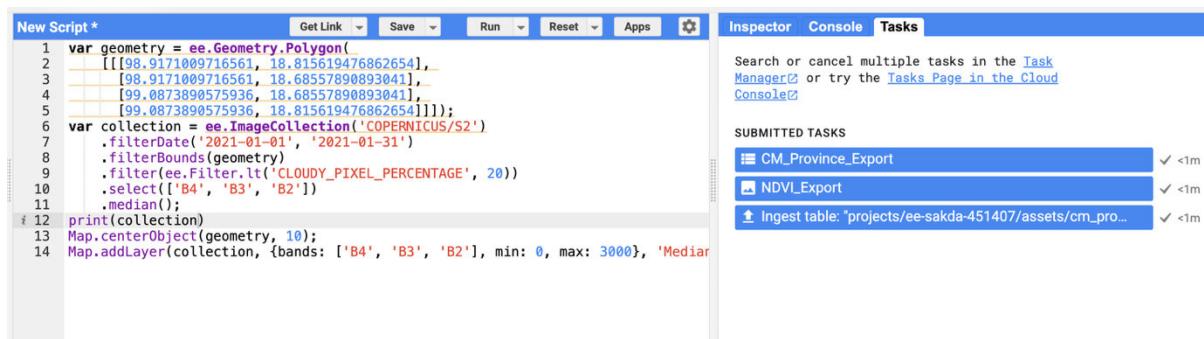
Inspector Console Tasks

- Point (99.0077, 18.7777) at 153m/px
- Pixels
 - Median Image: Image (3 bands)
 - B4: 919
 - B3: 1104
 - B2: 1308
- Objects
 - Median Image: Image (3 bands)
 - type: Image
 - bands: List (3 elements)
 - properties: Object (1 property)

2.2.6 Tasks

Tasks เป็นส่วนจัดการงานที่ส่งประมวลผล เป็นแท็บหนึ่งในส่วน Console (ด้านขวา) และบางครั้งอาจมีหน้าต่างแจ้งเตือนแยกต่างหาก

GEE ทำการประมวลผลบางอย่าง เช่น การ Export ข้อมูลขนาดใหญ่ การคำนวณที่ซับซ้อน ในลักษณะของ "Task" ซึ่งเป็นการทำงานแบบ Asynchronous (ทำงานเบื้องหลัง) ส่วน Tasks นี้จะแสดงรายการ Task ทั้งหมดที่ผู้ใช้งานได้ทำการสถานะของแต่ละ Task (เช่น Submitted, Running, Completed, Failed) และมีปุ่ม "Run" เพื่อเริ่มการทำงานของ Task ที่ยังไม่ได้เริ่ม การใช้งานเมื่อผู้ใช้งาน Export ข้อมูลไปยัง Google Drive หรือ Assets จะต้องเข้ามาที่ส่วน Tasks เพื่อกด "Run" ให้ Task นั้นเริ่มทำงานจริง และสามารถติดตามความคืบหน้าหรือตรวจสอบข้อผิดพลาดที่เกิดขึ้นระหว่างการ Export ได้จากส่วนนี้



```

New Script * Get Link Save Run Reset Apps
1 var geometry = ee.Geometry.Polygon(
2   [[[-98.9171009716561, 18.815619476862654], ->
3     [-98.9171009716561, 18.68557890893041], ->
4     [-99.0873890575936, 18.68557890893041], ->
5     [-99.0873890575936, 18.815619476862654]]]);
6 var collection = ee.ImageCollection('COPERNICUS/S2')
7 .filterDate('2021-01-01', '2021-01-31')
8 .filterBounds(geometry)
9 .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
10 .select(['B4', 'B3', 'B2'])
11 .median();
12 print(collection)
13 Map.centerObject(geometry, 10);
14 Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Median')

```

Inspector Console Tasks

Search or cancel multiple tasks in the [Task Manager](#) or try the [Tasks Page in the Cloud Console](#)

SUBMITTED TASKS

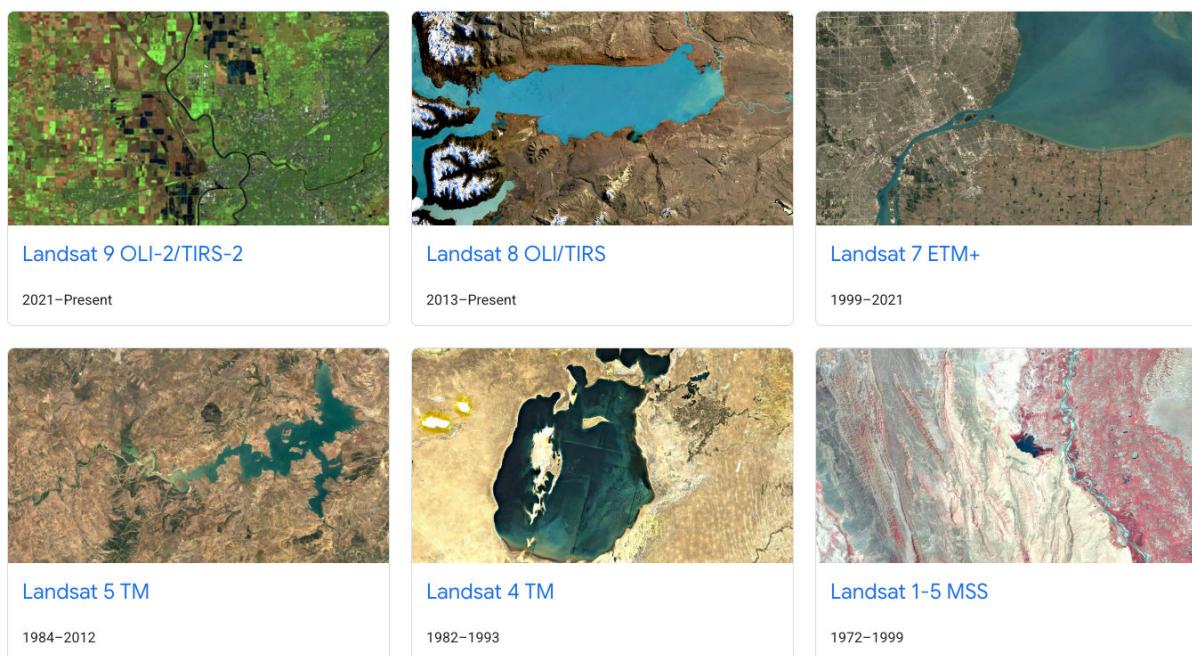
- CM_Province_Export ✓ <1m
- NDVI_Export ✓ <1m
- Ingest table: "projects/ee-sakda-451407/assets/cm_pro... ✓ <1m

บทที่ 3 ข้อมูลและแหล่งข้อมูลใน GEE

3.1 Imagery Dataset

Imagery คือชุดข้อมูลภาพที่ได้จากการเที่ยม อากาศยาน หรือเซนเซอร์เรดาร์ ซึ่งเป็นพื้นฐานสำคัญสำหรับการวิเคราะห์พื้นที่ต่าง ๆ ใน GEE ภาพเหล่านี้ถูกเก็บในรูปแบบแรสเตอร์ ที่แต่ละพิกเซลจะมีความเข้มแสงหรือดัชนีทางสเปกตรัม ชุดข้อมูลยอดนิยมได้แก่ ข้อมูลจากดาวเที่ยม Landsat Collection (ความละเอียดเชิงพื้นที่ 30 ม. ตั้งแต่ปี 1972 ถึงปัจจุบัน) ซึ่งเหมาะสมกับการศึกษาการเปลี่ยนแปลงภาคพื้นดินระยะยาว ข้อมูลจากดาวเที่ยม Sentinel-2 MSI (ความละเอียดเชิงพื้นที่ตั้งแต่ 10–20 ม. ตั้งแต่ปี 2015) ที่มีแบบอินฟราเรดใกล้ (Nir-Infrared) ซึ่งเหมาะสมกับการคำนวณ NDVI และ NDWI รวมถึง Sentinel-1 SAR (ความละเอียดเชิงพื้นที่ 10 ม. ตั้งแต่ปี 2014) ที่ส่งสัญญาณไมโครเวฟทะลุเมฆได้ดี ใช้สำหรับตรวจวัดความชื้นในดินหรือการเปลี่ยนแปลงโครงสร้างพื้นผิว หรือแม้แต่ภาพจากเครื่องบินหรือโดรนที่สามารถนำเข้าสู่ GEE เพื่อประมวลผลต่อได้

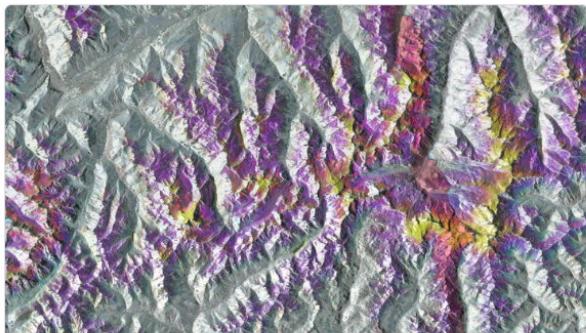
ตัวอย่างชุดข้อมูลจากดาวเที่ยม Landsat



ภาพที่ 5 ตัวอย่างชุดข้อมูลจากดาวเที่ยม Landsat

ที่มา <https://developers.google.com/earth-engine/datasets/catalog/landsat>

ຕ້ວຍ່າງຊຸດຂອ່ມູນຈາກດາວເຖິ່ນ Sentinel



Sentinel-1 SAR GRD: C-band Synthetic Aperture Radar

Data availability: 2014 – Present

The Sentinel-1 mission provides data from a dual-polarization C-band Synthetic Aperture Radar (SAR) instrument. SAR instruments are capable of acquiring meaningful data in all weather conditions (even clouds) during daytime and nighttime. Sentinel-1 data is used across many domains, including maritime activity, sea-ice mapping, humanitarian aid, crisis response, and forest management.



Sentinel-2 MSI: Multispectral Instrument

Data availability: 2015 – Present

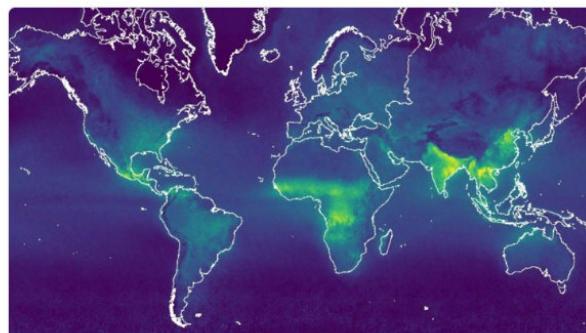
The Sentinel-2 mission collects high-resolution multispectral imagery useful for a broad range of applications, including monitoring of vegetation, soil and water cover, land cover change, as well as humanitarian and disaster risk.



Sentinel-3 OLCI EFR: Ocean and Land Color Instrument

Data availability: 2016 – Present

The Sentinel-3 instrument provides systematic measurements of the planet's oceans, land, ice, and atmosphere, including the temperature, color and height of the sea surface as well as the thickness of sea ice.



Sentinel-5P TROPOMI: TROPOspheric Monitoring Instrument

Data availability: 2018 – Present

The Sentinel-5 Precursor mission collects data useful for assessing air quality, including concentrations of: ozone, methane, formaldehyde, aerosol, carbon monoxide, nitrogen oxide, and sulphur dioxide.

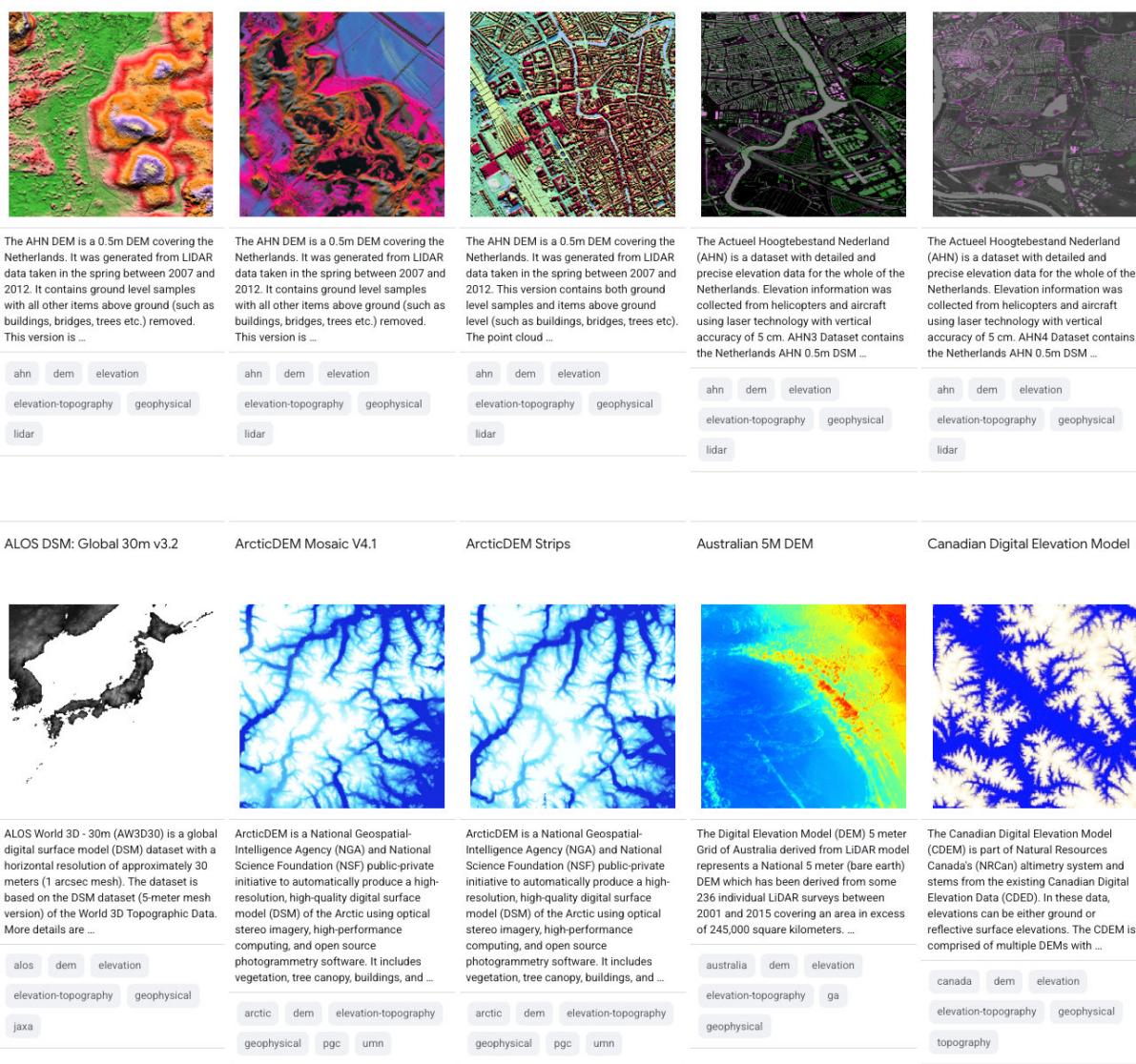
ກາພທີ່ 6 ຕ້ວຍ່າງຊຸດຂອ່ມູນຈາກດາວເຖິ່ນ Sentinel

ທີ່ມາ <https://developers.google.com/earth-engine/datasets/catalog/sentinel>

3.2 Terrain / Elevation Dataset

ข้อมูล Terrain/Elevation เป็นชุดข้อมูลระดับความสูงของพื้นผิวโลก (Digital Elevation Models: DEM) และข้อมูลอื่นๆ เช่น ความลาดเอียง (slope) หรือทิศรับแสง (aspect) ชุดข้อมูลที่ได้รับความนิยม ได้แก่ SRTM GL1 (ความละเอียดเชิงพื้นที่ 30 ม. จากปี 2000) ซึ่งให้ภาพรวมระดับความสูงระหว่างละติจูด -60° ถึง $+60^{\circ}$ ASTER GDEM (ความละเอียดเชิงพื้นที่ 30 ม. ครอบคลุมถึงขั้วโลก) และ HydroSHEDS/HydroLAKES สำหรับอุทกวิทยา ชุดข้อมูลเหล่านี้ช่วยให้สามารถวิเคราะห์การไหลของน้ำ จัดทำแผนที่พื้นที่เสี่ยงดินถล่ม หรือประเมินทิศทางการไหลของน้ำฝน

ตัวอย่างข้อมูลประเภท elevation-topography



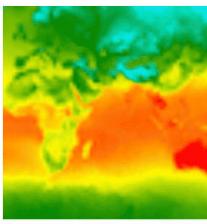
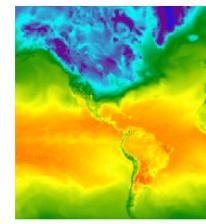
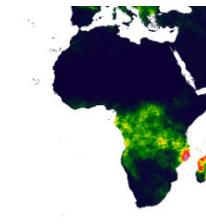
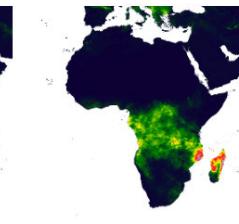
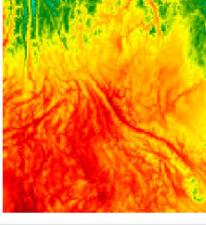
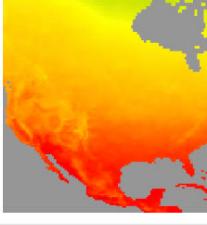
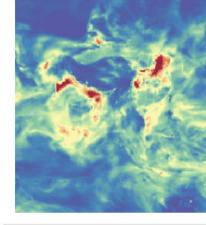
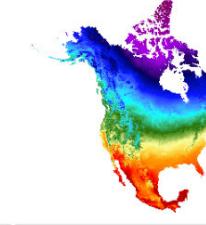
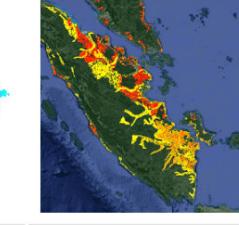
ภาพที่ 7 ตัวอย่างข้อมูลประเภท elevation-topography

ที่มา <https://developers.google.com/earth-engine/datasets/tags/elevation-topography>

3.3 Climate Dataset

ข้อมูล Climate ใน GEE หมายถึงชุดข้อมูลจำลองสภาพภูมิอากาศย้อนหลัง ซึ่งรวมรวมพารามิเตอร์ต่าง ๆ เช่น อุณหภูมิ ความกดอากาศ ปริมาณน้ำฝน ฯลฯ ตลอดช่วงหลายสิบปี ชุดข้อมูลที่นิยมได้แก่ NCEP/NCAR Reanalysis (ความละเอียดเชิงพื้นที่ 2.5 องศา ตั้งแต่ปี 1948) ERA5-Land (ความละเอียดเชิงพื้นที่ 9 กม. ตั้งแต่ปี 1981) ให้ข้อมูลเชิงลึกทั้งรายชั่วโมงและรายวันสำหรับงานเกษตรและอุตสาหกรรม และ GridMET (ความละเอียดเชิงพื้นที่ 4 กม. ตั้งแต่ปี 1979) ซึ่งเป็นชุดข้อมูลฝนและอุณหภูมิที่เหมาะสมกับการศึกษาโครงการปรับตัวทางการเกษตร

ตัวอย่างชุดข้อมูลประเภท Climate

Breathing Earth System Simulator (BESS) Radiation v1	CFSR: Climate Forecast System Reanalysis	CFSv2: NCEP Climate Forecast System Version 2, 6-Hourly Products	CHIRPS Daily: Climate Hazards Center InfraRed Precipitation With Station Data (Version 2.0 Final)	CHIRPS Pentad: Climate Hazards Center InfraRed Precipitation With Station Data (Version 2.0 Final)
				
Breathing Earth System Simulator (BESS) is a simplified process-based model that couples atmosphere and canopy radiative transfers, canopy photosynthesis, transpiration, and energy balance. It couples an atmospheric radiative transfer model and artificial neural network with forcings from MODIS atmospheric products to generate 5-km daily products. ...	The National Centers for Environmental Prediction (NCEP) Climate Forecast System Reanalysis (CFSR) was designed and executed as a global, high-resolution, coupled atmosphere-ocean-land-surface-sea ice system to provide the best estimate of the state of these coupled domains over the 32-year period of record from January ...	The National Centers for Environmental Prediction (NCEP) Climate Forecast System (CFS) is a fully coupled model representing the interaction between the Earth's atmosphere, oceans, land, and sea ice. CFS was developed at the Environmental Modeling Center (EMC) at NCEP. The operational CFS was upgraded to ...	Climate Hazards Center InfraRed Precipitation with Station data (CHIRPS) is a 30+ year quasi-global rainfall dataset. CHIRPS incorporates 0.05° resolution satellite imagery with in-situ station data to create gridded rainfall time series for trend analysis and seasonal drought monitoring.	Climate Hazards Center InfraRed Precipitation with Station data (CHIRPS) is a 30+ year quasi-global rainfall dataset. CHIRPS incorporates 0.05° resolution satellite imagery with in-situ station data to create gridded rainfall time series for trend analysis and seasonal drought monitoring.
climate evapotranspiration gpp modis-derived par radiation	climate daylight flux forecast geophysical ncep	climate daylight flux forecast geophysical ncep	chrg climate geophysical precipitation ucsb weather	chrg climate geophysical precipitation ucsb weather
CHIRTS-daily: Climate Hazards Center InfraRed Temperature with Stations daily temperature data	CPC Global Unified Temperature	Copernicus Atmosphere Monitoring Service (CAMS) Global Near-Real-Time	Daymet V4: Daily Surface Weather and Climatological Summaries	Drained Organic Soils Emissions (Annual) 1.0
				
The Climate Hazards Center InfraRed Temperature with Stations daily temperature data product (CHIRTS-daily; Verdin et al. 2020) is a quasi global, high-resolution gridded dataset ($0.05^{\circ} \times 0.05^{\circ}$ resolution, 60°S - 70°N) that provides daily minimum (T_{min}) and maximum 2-meter temperatures (T_{max}) and four derived variables: saturation vapor ...	This dataset provides a gridded analysis of daily surface air temperature over global land areas, including daily maximum (T_{max}), minimum (T_{min}) temperatures. Spanning from 1979 to the present, the data is presented on 0.5-degree latitude/longitude grids, aligning with the resolution of CPC's gauge-based global daily ...	The Copernicus Atmosphere Monitoring Service provides the capacity to continuously monitor the composition of the Earth's atmosphere at global and regional scales. The main global near-real-time production system is a data assimilation and forecasting suite providing two 5-day forecasts per day for aerosols and chemical ...	Daymet V4 provides gridded estimates of daily weather parameters for Continental North America, Hawaii, and Puerto Rico (Data for Puerto Rico is available starting in 1950). It is derived from selected meteorological station data and various supporting data sources. Compared to the previous version, Daymet ...	The two related FAO datasets on Drained Organic Soils provide estimates of: DROSA-A: area of Organic Soils (in hectares) drained for agricultural activities (cropland and grazed grassland) DROSE-A: carbon (C) and nitrous oxide (N2O) estimates (in gigagrams) from the agricultural drainage of organic soils under ...
chrg climate daily era5 geophysical reanalysis	climate daily noaa precipitation weather	aerosol atmosphere climate copernicus ecmwf forecast	climate daily daylight flux geophysical nasa	agriculture climate climate-change emissions fao

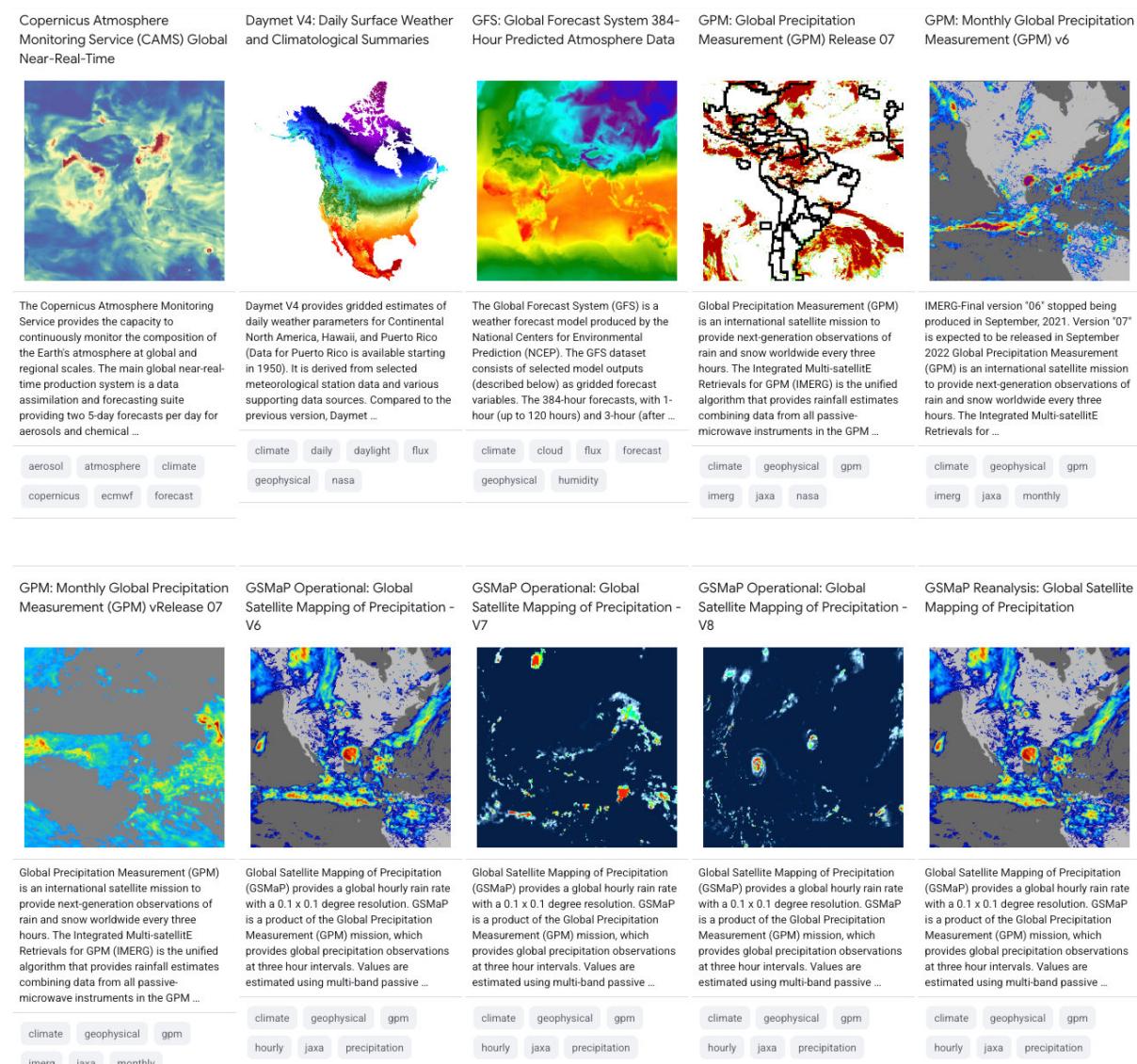
ภาพที่ 8 ตัวอย่างของข้อมูลประเภท Climate

ที่มา <https://developers.google.com/earth-engine/datasets/tags/climate>

3.4 Weather Dataset

ข้อมูล Weather เป็นชุดข้อมูลสภาพอากาศบังคับและการพยากรณ์ระยะสั้น เช่น ฝน ลม อุณหภูมิที่อัปเดตรายวันหรือรายชั่วโมง ตัวอย่างเช่น CHIRPS (ความลักษณะเชิงพื้นที่ 0.05องศา เป็นข้อมูลรายวัน ตั้งแต่ปี 1981) สำหรับการติดตามฝนตกและน้ำท่วม NOAA GFS (ความลักษณะเชิงพื้นที่ 0.25 องศา ทุก 6 ชั่วโมง) สำหรับพยากรณ์สภาพอากาศ และ TRMM/GPM (ความลักษณะเชิงพื้นที่ 0.1 องศา รายชั่วโมง) ที่เหมาะสมกับพื้นที่ในเขตร้อน ชุดข้อมูลเหล่านี้ใช้ในงานเดือนภัยน้ำท่วม การวางแผนการเกษตร และระบบสนับสนุนการตัดสินใจ

ตัวอย่างชุดข้อมูลประเภท Weather



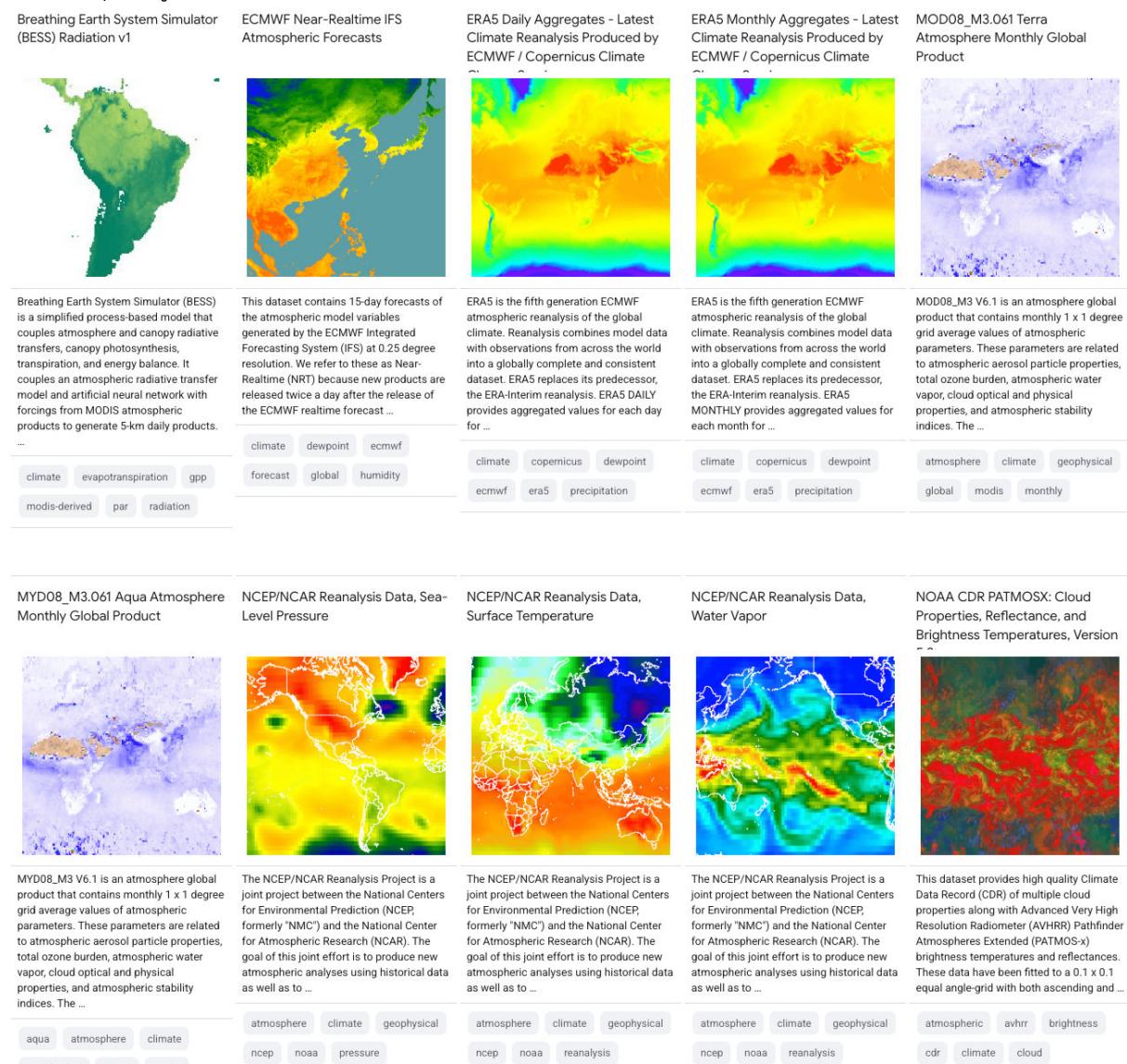
ภาพที่ 9 ตัวอย่างชุดข้อมูลประเภท Weather

ที่มา <https://developers.google.com/earth-engine/datasets/tags/weather>

3.5 Atmospheric Dataset

ข้อมูล Atmospheric เป็นข้อมูลบรรยายการซั่นต่าง ๆ ตั้งแต่โอดีตจนถึงปัจจุบันของ ชุดข้อมูลเช่น MODIS Atmospheric Products (ความละเอียดเชิงพื้นที่ 1 กม.) ให้ค่า Aerosol Optical Depth, Water Vapor และ Cloud Fraction, OMI/TROPOMI (ความละเอียดเชิงพื้นที่ 13×24 กม.) สำหรับการสังเกตโอดีตรายวัน MERRA-2 Reanalysis (ความละเอียดเชิงพื้นที่ 0.5 องศา รายชั่วโมง) ที่ให้ข้อมูลกากเรื่องผลกระทบและผู้ล่อง ซึ่งข้อมูลเหล่านี้จำเป็นสำหรับงานวิจัยสภาพอากาศระดับภูมิภาคและการประเมินคุณภาพอากาศ

ตัวอย่างชุดข้อมูลประเภท Atmospheric



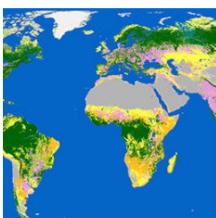
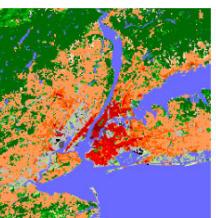
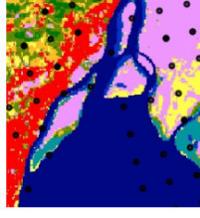
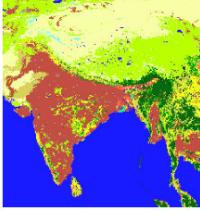
ภาพที่ 10 ตัวอย่างชุดข้อมูลประเภท Weather

ที่มา <https://developers.google.com/earth-engine/datasets/tags/atmospheric>

3.6 Land Cover / Land Use Dataset

ข้อมูล Land Cover/Land Use เป็นข้อมูลจำแนกประเภทพื้นที่ตามการปักคลุมหรือลักษณะการใช้ประโยชน์ ชุดข้อมูลสำคัญ เช่น MODIS MCD12Q1 (500 ม. รายปี แผนที่ 17 ชั้น) สำหรับจำแนกเช่น ป่าเมือง เกษตร ESA CCI (300 ม. รายปี) มีความละเอียดสูงเฉพาะโครงการ Climate Change Initiative, Hansen Global Forest Change (ความละเอียดเชิงพื้นที่ 30 ม. รายปี) สำหรับติดตามการตัดไม้ทำลายป่าและการฟื้นฟูป่า ข้อมูลเหล่านี้ช่วยให้ประเมินการเปลี่ยนแปลงที่เดินในระดับโลก

ตัวอย่างข้อมูลประเภท Land Cover / Land Use

ESA WorldCover 10m v100	ESA WorldCover 10m v200	GlobCover: Global Land Cover Map	Global map of Local Climate Zones, latest version	Google Global Landsat-based CCDC Segments (1999-2019)
				
The European Space Agency (ESA) WorldCover 10 m 2020 product provides a global land cover map for 2020 at 10 m resolution based on Sentinel-1 and Sentinel-2 data. The WorldCover product comes with 11 land cover classes and has been generated in the framework of ...	The European Space Agency (ESA) WorldCover 10 m 2021 product provides a global land cover map for 2021 at 10 m resolution based on Sentinel-1 and Sentinel-2 data. The WorldCover product comes with 11 land cover classes and has been generated in the framework of ...	GlobCover 2009 is a global land cover map based on ENVISAT's Medium Resolution Imaging Spectrometer (MERIS) Level 1B data acquired in full resolution mode with a spatial resolution of approximately 300 meters.	Since their introduction in 2012, Local Climate Zones (LCZs) emerged as a new standard for characterizing urban landscapes, providing a holistic classification approach that takes into account micro-scale land-cover and associated physical properties. This global map of Local Climate Zones, at 100m pixel size and ...	This collection contains precomputed results from running the Continuous Change Detection and Classification (CCDC) algorithm on 20 years of Landsat surface reflectance data. CCDC is a breakpoint finding algorithm that uses harmonic fitting with a dynamic RMSE threshold to detect breakpoints in time-series data. The ...
esa landcover landuse landuse-landcover sentinel1-derived sentinel2-derived	esa landcover landuse landuse-landcover sentinel1-derived sentinel2-derived	esa landcover landuse-landcover	climate landcover landuse-landcover urban	change-detection google landcover landsat-derived landuse landuse-landcover
Iran Land Cover Map v1 13-class (2017)	LUCAS Copernicus (Polygons with attributes, 2018) V1	LUCAS Harmonized (Theoretical Location, 2006–2018) V1	Land Cover of North America at 30 meters, 2020	MCD12C1.061 MODIS Land Cover Type Yearly Global 0.05 Deg CMG
				
The Iran-wide land cover map was generated by processing Sentinel imagery within the Google Earth Engine Cloud platform. For this purpose, over 2,500 Sentinel-1 and over 11,000 Sentinel-2 images were processed to produce a single mosaic dataset for the year 2017. Then, an object-based Random ...	The Land Use/Cover Area frame Survey (LUCAS) in the European Union (EU) was set up to provide statistical information. It represents a triennial in-situ landcover and land-use data-collection exercise that extends over the whole of the EU's territory. LUCAS collects information on land cover and ...	The Land Use/Cover Area frame Survey (LUCAS) in the European Union (EU) was set up to provide statistical information. It represents a triennial in-situ landcover and land-use data-collection exercise that extends over the whole of the EU's territory. LUCAS collects information on land cover and ...	The 2020 North American Land Cover 30-meter dataset was produced as part of the North American Land Change Monitoring System (NALCMS), a trilateral effort between Natural Resources Canada, the United States Geological Survey, and three Mexican organizations including the National Institute of Statistics and Geography ...	The Terra and Aqua combined Moderate Resolution Imaging Spectroradiometer (MODIS) Land Cover Climate Modeling Grid (CMG) (MCD12C1) Version 6.1 data product provides a spatially aggregated and reprojected version of the tiled MCD12Q1 Version 6.1 data product. Maps of the International Geosphere-Biosphere Programme (IGBP), University of ...
landcover landuse-landcover	copernicus eu jrc landcover landuse landuse-landcover	eu jrc landcover landuse landuse-landcover lucas	landcover landsat landuse-landcover nlcd	landcover landuse-landcover modis nasa usgs yearly

ภาพที่ 11 ตัวอย่างชุดข้อมูลประเภท Land Cover / Land Use

ที่มา <https://developers.google.com/earth-engine/datasets/tags/landuse-landcover>

3.7 Vegetation Indices Dataset

ข้อมูล Vegetation Indices คือผลลัพธ์ที่คำนวณจากภาพดาวเทียมเพื่อประเมินสภาพพืชพรรณ ซึ่ดข้อมูลเช่น MODIS MOD13Q1 NDVI/EVI (ความละเอียดเชิงพื้นที่ 250 ม. ทุก 16 วัน) ให้ภาพรวมสุภาพพืช, VIIRS VNP13A1 (ความละเอียดเชิงพื้นที่ 500 ม. ทุก 16 วัน) ที่เพิ่มความละเอียดและลดการส่องลับ หมายเหตุ การติดตามการเติบโตของพืชในระดับภูมิภาคถึงระดับท้องถิ่น ซึ่ดดัชนีเหล่านี้ใช้ในงานเกษตรแม่นยำและการประเมินผลกระทบสภาพอากาศ

3.8 Hydrology Dataset

ข้อมูล Hydrology เป็นการศึกษาน้ำผิวดิน แม่น้ำ ทะเลสาบ และการเคลื่อนไหวของน้ำ ซึ่ดข้อมูลที่สำคัญได้แก่ JRC Global Surface Water (30 ม. รายปี) สำหรับแผนที่น้ำผิวดินและการเปลี่ยนแปลงพื้นที่น้ำ, GRWL Global River Widths (30 ม.) ให้ความกว้างแม่น้ำทั่วโลก SWOT (upcoming) ซึ่งให้ข้อมูลสมุทรศาสตร์ความละเอียดสูง ข้อมูลเหล่านี้จำเป็นสำหรับการจัดการทรัพยากรน้ำและประเมินความเสี่ยงน้ำท่วม

3.9 Soils Dataset

ข้อมูล Soils ให้ข้อมูลลักษณะทางเคมีและกายภาพของดิน เช่น organic carbon, texture, pH ซึ่ดข้อมูลเช่น SoilGrids by ISRIC (ความละเอียดเชิงพื้นที่ 250 ม. หลายชั้นความลึก) ให้รายละเอียดหลายระดับลึก, Harmonized World Soil Database (HWSD, 1 กม.) รวบรวมแหล่งข้อมูลเพื่อสร้างแผนที่ดินระดับโลก ข้อมูลดินเหล่านี้ใช้ประเมินการเก็บกักคาร์บอนในดินและการจัดการเกษตร

3.10 Fire

ข้อมูล Fire เก็บข้อมูลจุดความร้อนและไฟป่าที่ตรวจจับจากดาวเทียม ซึ่ดข้อมูลเช่น MODIS Active Fire (MCD14DL, 1 กม. รายวัน) และ VIIRS Active Fire (VNP14IMGML, 375 ม. รายวัน) ที่ตรวจจับไฟขนาดเล็กกว่าเดิม ข้อมูลนี้ใช้สำหรับเฝ้าระวังไฟป่า การประเมินพื้นที่เสี่ยหาย และวางแผนป้องกันเพลิงไหม้

3.11 Nighttime Lights

ข้อมูล Nighttime Lights คือภาพการส่องสว่างในยามค่ำคืน ที่สะท้อนกิจกรรมมนุษย์และการกระจายตัวของเมือง ซึ่ดข้อมูลเช่น DMSP-OLS (ความละเอียดเชิงพื้นที่ 1 กม. รายปี ย้อนกลับถึงปี 1992) และ VIIRS

Day/Night Band (500 ม. รายเดือน) ถูกนำไปใช้ในการเติบโตเมือง ประเมินเศรษฐกิจ และติดตามการใช้พลังงาน

3.12 Socioeconomic Dataset

ข้อมูล Socioeconomic เป็นข้อมูลประชากรและสถิติทางเศรษฐกิจ เช่น WorldPop Population (100 ม. รายปี) สำหรับการกระจายตัวประชากร, Gridded GDP (ความลับเฉียงพื้นที่ 5 กม. รายปี) สำหรับวิเคราะห์ GDP ต่อพื้นที่, LandScan Global (ความลับเฉียงพื้นที่ 1 กม. รายชั่วโมง) ให้การจำลองการกระจายประชากรตามกิจกรรม อำนวยความสะดวกในการวางแผนเมืองและนโยบายสาธารณะ

3.13 Oceanography Dataset

ข้อมูล Oceanography เป็นข้อมูลสมุทรศาสตร์ เช่น อุณหภูมิน้ำคลื่น และสารอินทรีย์ในน้ำ ชุดข้อมูล เช่น AVHRR Pathfinder SST (4 กม. รายวัน ตั้งแต่ปี 1981), Copernicus Marine OSTIA (1 กม.) สำหรับพยากรณ์อุณหภูมิและความเค็ม, GlobColour Chlorophyll-a (4 กม. รายเดือน) สำหรับประเมินความอุดมสมบูรณ์ในมหาสมุทร ชุดข้อมูลนี้จำเป็นสำหรับการศึกษาระบบนิเวศทางทะเลและการจัดการประมง

3.14 Infrastructure Dataset

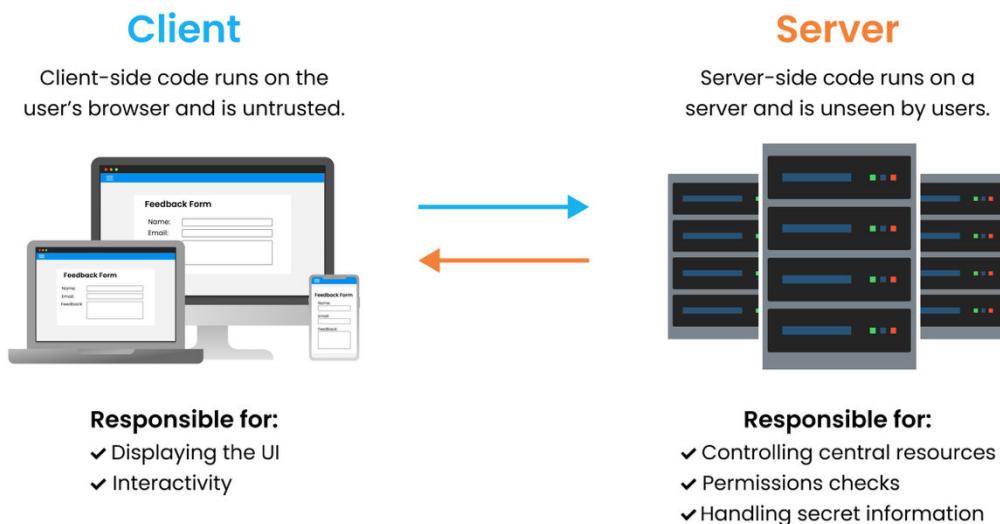
ข้อมูล Infrastructure เป็นการรวบรวมข้อมูลถนน อาคาร และโครงสร้างพื้นฐานจากภาครัฐหรือชุมชน ชุดข้อมูล เช่น OpenStreetMap (vector อัปเดตต่อเนื่อง), GRIP – Global Roads Inventory (100 ม.) ให้เครือข่ายถนนหลัก, Microsoft Building Footprints สำหรับรูปร่างอาคารที่สร้างด้วย AI ข้อมูลเหล่านี้ช่วยพัฒนาโมเดลขนาดใหญ่ ออกแบบเมืองอัจฉริยะ และวิเคราะห์พื้นที่ที่มีโครงสร้างพื้นฐานหนาแน่น

บทที่ 4 พื้นฐานภาษา JavaScript สำหรับ GEE

GEE มีลักษณะการทำงานแบบเป็น Client-side และ Server-side ที่เราสามารถเขียนภาษา JavaScript ใน Code Editor สั่งงานไปยัง Server ได้ จะเห็นได้ว่าเป็นภาษา JavaScript เป็นภาษาที่สำคัญในการเขียน скриปต์ผ่าน Code Editor แม้ว่าผู้ใช้ไม่จำเป็นต้องเป็นผู้เขียนภาษา JavaScript แต่การทำความเข้าใจพื้นฐานบางประการจะช่วยให้เรียนรู้ GEE ได้รวดเร็วยิ่งขึ้น

4.1 Server-side vs Client-side

ใน GEE การประมวลผลข้อมูลทั้งหมดจะเกิดขึ้นบน เชิร์ฟเวอร์ (Server-side) ของ Google แต่เราสามารถสั่งงานและดึงผลลัพธ์มาแสดงบน ไคลเอ็นต์ (Client-side) ของเราเองได้ แนวคิดนี้สำคัญต่อการเขียน скриปต์ GEE ให้มีประสิทธิภาพและหลีกเลี่ยงการร้องขอข้อมูลจำนวนมากไปยังเชิร์ฟเวอร์โดยไม่จำเป็น ซึ่งทั้ง Server-side และ Client-side มีบทบาทและข้อจำกัดที่แตกต่างกัน ดังนี้



ภาพที่ 12 การทำงานของ Server-side vs Client-side

ที่มา: <https://anvil.works/articles/img/client-vs-server/client-server-architecture.png>

Server-side

คำสั่ง Server-side เป็นออบเจกต์ที่อยู่ภายใต้ namespace ee เช่น ee.Image, ee.ImageCollection, ee.Geometry เป็นต้น รันโค้ดบนโครงสร้างพื้นฐานคลาวด์ของ Google การทำงานของฝั่ง Server-side เมื่อเขียนโค้ดเรียกเมธอด (filterDate(), map(), reduceRegion() ฯลฯ) ระบบจะไม่ประมวลผลทันที แต่จะสร้าง “task” หรือคำสั่ง (computation graph) ค้างไว้ ก่อนจะรันจริง โดย GEE จะรวมรวม task ทั้งหมดมาปรับ

ลำดับขั้นตอน และแบ่งงานไปยังเซิร์ฟเวอร์หลายเครื่อง (parallel processing) เพื่อเพิ่มประสิทธิภาพของการทำงาน ผลลัพธ์จะยังไม่กลับมาที่ฝั่ง JavaScript code จนกว่าจะเรียกใช้ method ที่ดึงข้อมูล ซึ่งข้อดีของหลักการนี้คือ ช่วยให้การประมวลผลข้อมูลขนาดใหญ่ (petabytes) ทำได้รวดเร็ว ไม่ต้องกังวลเรื่องขนาดแวร์ หรือ storage ของผู้ใช้ แต่ก็มีข้อจำกัดคือไม่สามารถทำ loop ที่ขึ้นกับค่าผลลัพธ์ก่อนหน้าได้โดยตรง (เพราะยังไม่ได้ประมวลผล) ซึ่งต้องระวังการเรียกบ่อยเกินไป เพราะจะทำให้เกิด latency สูงและจำกัดปริมาณการดึงข้อมูล

Client-side

เป็น JavaScript code ที่รันบนเบราว์เซอร์ของผู้ใช้ ทำหน้าที่ควบคุม logic ของ UI การส่งคำขอไปยังฝั่ง server และการจัดการผลลัพธ์ โดยตัวอย่างการดึงผลลัพธ์จาก Server เช่น การใช้ .getInfo()

```
var meanDict = ndviImage.reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: roi,
  scale: 10
}).getInfo();
print('Mean NDVI:', meanDict.NDVI);
```

ตารางที่ 1 สรุปความแตกต่างของ Server-side และ Client-side

การทำงาน	Server-side	Client-side
สถานที่รันโค้ด	คลาวด์ GEE	Browser ของผู้ใช้
ชนิดของเจกเตอร์	ee.Image, ee.FeatureCollection ฯลฯ	JavaScript primitive (Number, Object, Array)
การประมวลผล	Lazy evaluation; queue & parallel execution	ดึงผลลัพธ์แล้วจัดการต่อใน UI หรือ logic ของคุณ
วิธีดึงผลลัพธ์	-	.getInfo(), .evaluate()
ข้อจำกัดหลัก	ไม่รุ่งผลทันที ต้องรอการประมวลผล และดึงข้อมูล	ต้องระวัง latency และ quota ของ API

4.2 ตัวแปร (Variables)

การประกาศตัวแปร ใน JavaScript (และ GEE) จะใช้คำว่า var เพื่อประกาศตัวแปรสำหรับเก็บค่าต่างๆ source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter4.md#1-variables-and-data-types

```
// Variable declaration
var number = 42;                      // Number
var text = "Hello, Earth!";              // String
var boolean = true;                     // Boolean
var list = [1, 2, 3, 4];                 // Array
var object = {key: "value"};             // Object
```

การตั้งชื่อตัวแปร

- ต้องขึ้นต้นด้วยตัวอักษร (a-z, A-Z), _ หรือ \$
- ห้ามใช้ตัวเลขขึ้นต้น เช่น 1value ไม่ได้ แต่ value1 ได้
- แยกคำด้วย camelCase: myVariableName
- หลีกเลี่ยงใช้ชื่อที่ชนกับคำส่วน (reserved words) เช่น for, function และ

*หมายเหตุ ใน GEE ยังไม่รองรับการประกาศตัวแปรรูปแบบใหม่ เช่น let หรือ const

4.3 ประเภทข้อมูล (Data Types)

JavaScript รองรับการจัดเก็บและประมวลผลข้อมูลหลายรูปแบบ ซึ่งแบ่งออกได้เป็นสองกลุ่มใหญ่คือ

Primitive Types และ **Reference Types** โดยมีรายละเอียดดังนี้

- Primitive Types ประเภทข้อมูลพื้นฐานที่เก็บเพียงค่าตัวเดียว เช่น Number, String, Boolean
- Reference Types ประเภทข้อมูลเหล่านี้เป็นวัตถุ (objects) ที่เก็บข้อมูลในหน่วยความจำเป็นโครงสร้าง และตัวแปรจะอิงถึงตำแหน่งที่เก็บข้อมูลนั้น เช่น Array, Object, Date

ตารางที่ 2 ประเภทข้อมูลพื้นฐานใน JavaScript

ประเภท	คำอธิบาย	ตัวอย่าง
Number	ตัวเลขทั้งเต็มและทศนิยม	42, 3.14
String	ข้อความ	"Hello, GEE", 'NDVI คืออะไร?'
Boolean	ค่าทางตรรกะ จริง/เท็จ	true, false
Array	อาร์เรย์	[1, 2, 3, 'a', true]
Object	อ็อบเจกต์	{name: 'Thailand', area: 513120};
Date	วันที่และเวลา	"2025-04-01T12:30:00Z"

อาเรย์คือโครงสร้างข้อมูลแบบลำดับ (ordered list) ที่เก็บค่าห่างๆ ค่าวาด้วยกัน

```
// Array variable
var array = [1, 2, 3, 4, 5];
// Accessing array elements
var firstElement = array[0];      // 1
// Modifying array elements
array[1] = 10;                  // [1, 10, 3, 4, 5]
```

อ็อบเจกต์ใน JavaScript คือชุดข้อมูลแบบ key-value (เหมือน JSON)

```
// Object variable
var obj = {name: "Earth", age: 4.5};
obj.name = "Mars";                // {name: "Mars", age: 4.5}
// Accessing object properties
var name = obj.name;             // "Mars"
// Modifying object properties
obj.age = 4.6;                  // {name: "Mars", age: 4.6}
```

GEE Object เป็นอ็อบเจกต์ (Objects) แบบหนึ่งที่ใช้แทนโครงสร้างข้อมูลภูมิสารสนเทศ เช่น ee.Image, ee.ImageCollection, ee.Feature, ee.FeatureCollection, ee.Geometry, ee.Reducer, ee.Filter, ee.List, ee.Dictionary

```
// Earth Engine objects
var numList = ee.List([1, 2, 3, 4, 5]);
var image = ee.Image("LANDSAT/LC08/C01/T1/LC08_044034_20140318");
var geometry = ee.Geometry.Point([-122.082, 37.42]);
```

4.4 การ Comments

การเขียน Comment ในโค้ดเป็นสิ่งที่ดีมาก ช่วยให้อธิบายว่าโค้ดส่วนนั้นๆ ทำงานอย่างไร มีประโยชน์ทั้งต่อตนเองในอนาคตเมื่อกลับมาอ่านโค้ด และต่อผู้อื่นที่อาจนำโค้ดของเราไปใช้หรือพัฒนาต่อ GEE จะไม่ประมวลผลส่วนที่เป็น Comment รูปแบบการเขียน Comment มีดังนี้

การ Comment บรรทัดเดียว (Single-line comment) โดยใช้เครื่องหมาย // นำหน้าข้อความ Comment นั้นจะครอบคลุมเฉพาะส่วนที่อยู่หลัง // ในบรรทัดนั้น

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter4.md#2-comment

```
// Single line comment
```

การ Comment หลายบรรทัด (Multi-line comment) ใช้เครื่องหมาย `/*` เพื่อเริ่มต้น Comment และ `*/` เพื่อสิ้นสุด Comment ข้อความทั้งหมดที่อยู่ระหว่างเครื่องหมายนี้จะถือเป็น Comment

```
/*
Multi-line comment
    This is a multi-line comment
        that spans multiple lines.
*/
```

4.5 พังก์ชัน (Functions)

การเขียนพังก์ชันต้องระบุคำว่า function และระบุชื่อให้กับพังก์ชัน ก่อนปิดด้วย ()

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter4.md#3-functions

```
// function declaration
function showMessage() {
    print('Hello, Earth Engine!');
}

// call the function
showMessage();

// function with parameters
function addNumbers(a, b) {
    return a + b;
}
var sum = addNumbers(5, 10);
```

โดยที่

Parameters ชื่อตัวแปรในวงเล็บ คือค่าที่จะส่งเข้าไป

Return เป็นคำสั่งส่งค่ากลับออกจากพังก์ชัน ถ้าไม่มี return จะคืนค่าเป็น undefined

GEE มีพังก์ชันสำหรับรูปจำนวนมากสำหรับจัดการและวิเคราะห์ข้อมูล การเรียกใช้พังก์ชันทำได้โดยการใส่ชื่อ พังก์ชันตามด้วยวงเล็บ () และอาจมีพารามิเตอร์ (arguments) อยู่ในวงเล็บ

```
// Earth Engine function
var roi = ee.Geometry.Polygon(
    [[[98.9171009716561, 18.815619476862654],
      [98.9171009716561, 18.68557890893041],
      [99.0873890575936, 18.68557890893041],
      [99.0873890575936, 18.815619476862654]]]);
// Define a function to calculate NDVI for one image
```

```

function calcNDVI(image) {
    // Compute normalized difference of bands B8 and B4
    return image.normalizedDifference(['B8', 'B4'])
        .rename('NDVI');
}

// Apply the function to every image in the collection
var collection = ee.ImageCollection('COPERNICUS/S2')
    .filterDate('2021-01-01', '2021-01-31')
    .filterBounds(roi);

var ndviCollection = collection.map(calcNDVI);

// Compute the median composite of NDVI
var medianNDVI = ndviCollection.median();

Map.addLayer(medianNDVI, {min: 0, max: 1}, 'Median NDVI');

```

4.6 การใช้ if

if...else เป็นคำสั่งการทำงานตามเงื่อนไขที่กำหนด โดยใน client-side จะใช้ if ในปริญบที่บค่าที่เป็นตัวแปร JavaScript ปกติ (primitive)

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter4.md#4-ifelse-statements

```

// Client-side if...else
let x = 7;
let y = 5;

// Simple if-else to compare JS numbers
if (x > y) {
  print('x is greater than y'); // prints: x is greater than y
} else if (x === y) {
  print('x is equal to y');
} else {
  print('x is less than y');
}

```

ขณะที่ server-side ต้องใช้ ee.Algorithms มาใช้แทน

```

// Earth Engine if...else
var image = ee.Image('LANDSAT/LC08/C01/T1/LC08_044034_20140318');
var ndvi = image.normalizedDifference(['B5', 'B4']);
var threshold = 0.5;
var mask = ndvi.gt(threshold);
var maskedImage = image.updateMask(mask);

```

```
Map.addLayer(maskedImage, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Masked Image');

// ee.Algorithms.If
var condition = ee.Number(5);
var result = ee.Algorithms.If(condition.gt(0), 'Positive', 'Negative');
print('Result:', result); // prints: Result: Positive
```

4.7 การใช้งาน Loop

loop เป็นการทําซําๆวายให้เราประมวลผลชุดข้อมูลหรือลิสต์จาก client-side

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter4.md#5-loops

```
// Client-side for loop
for (var i = 0; i < 5; i++) {
    print('Iteration:', i);
}

// Client-side while loop
var j = 0;
while (j < 5) {
    print('While loop iteration:', j);
    j++;
}

// map function
var numbers = [1, 2, 3, 4, 5];
var squaredNumbers = numbers.map(function(num) {
    return num * num;
});
print('Squared Numbers:', squaredNumbers); // [1, 4, 9, 16, 25]
```

ขณะที่ผู้ที่ server-side จะเป็นเรียกข้อมูลที่ลิสตรายการบน server 'ไม่ได้ดึงข้อมูลทั้งหมดมาที่ client เพื่อวน loop

```
// Server-side for loop
var serverList = ee.List([1, 2, 3, 4, 5]);
var serverSquared = serverList.map(function(num) {
    return ee.Number(num).multiply(ee.Number(num));
});
print('Server Squared:', serverSquared); // [1, 4, 9, 16, 25]

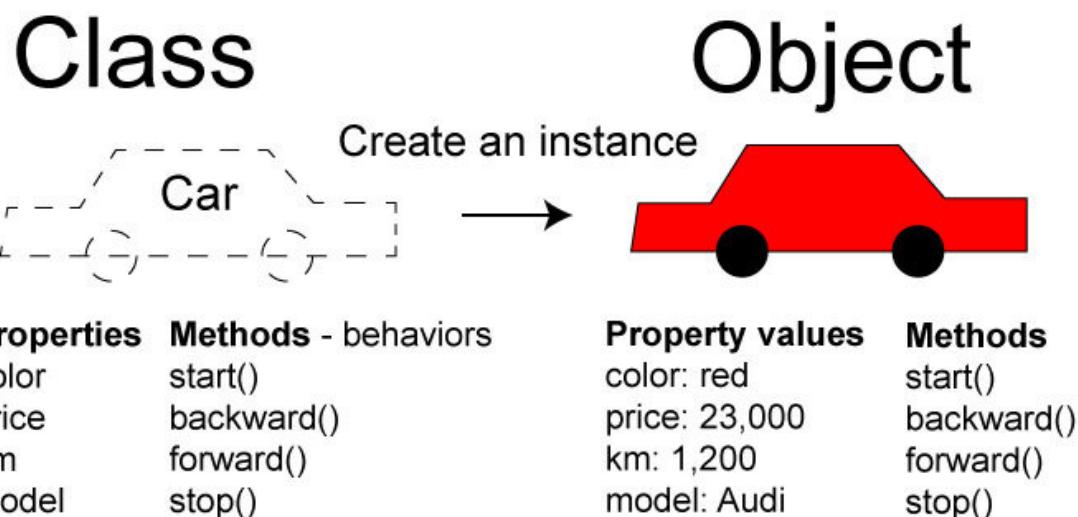
// Earth Engine map function
var collection = ee.ImageCollection('LANDSAT/LC08/C01/T1');
var ndviCollection = collection.map(function(image) {
    return image.normalizedDifference(['B5', 'B4']).rename('NDVI');
```

```
});
```

ในฝั่ง Client-side loops (for, while) เหมาะกับ data โครงสร้าง JavaScript เล็กๆ ในฝั่ง Server-side mapping (ee.List.map, ee.ImageCollection.map) สำหรับข้อมูลขนาดใหญ่ใน GEE ควรหลีกเลี่ยงการใช้ client-side loop กับ ee.ImageCollection เพราะอาจทำงานช้าและเกิน quota ได้

4.8 แนวคิดการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming: OOP)

การเขียนโปรแกรมเชิงวัตถุ (OOP) เป็นแนวคิดในการออกแบบและพัฒนาโปรแกรมโดยใช้ “วัตถุ” (objects) เป็นหน่วยย่อยหลักแทนการเขียนโปรแกรมเชิงกระบวนการ (procedural programming) ซึ่งมีจุดเด่นอยู่ที่ การกำหนดคุณลักษณะ (attributes) และพฤติกรรม (methods) ไว้ด้วยกันในโครงสร้างเดียว ทำให้โค้ดอ่านง่าย ดูแลรักษาง่าย และนำกลับมาใช้ซ้ำได้ยั่งยืน



ภาพที่ 13 แนวคิดการเชิงวัตถุ
ที่มา <https://www.youtube.com/watch?v=JuXRKGjDygs>

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter4.md#6-object

```
var Car = {
  wheels: 4,
  door: 2,
  start: function() {
    print('Car started');
    return this;
}
```

```

};

var tota = Car;
print('Toyota wheels:', tota.wheels);
tota.color = "red";
print('Tota color:', tota.color);
tota.start();

var hoda = Car;
hoda.door = 5;
print('Hoda door:', hoda.door);

tota.drive = function() {
  print('Car is driving');
  return this;
};
// tota.drive();

tota.stop = function() {
  print('Car stopped');
  return this;
};
// tota.stop();
// chain method calls
tota.start().drive().stop();

```

โดยใน GEE ได้ใช้แนวคิดพื้นฐานของ OOP เช่น คลาส (Class), ออบเจกต์ (Object/Instance), เมธอด (Method) และ พร็อพเพอร์ตี้ (Property) มาใช้ ดังนี้

Class เป็นแบบพิมพ์เขียวหรือแม่แบบ (blueprint) ที่กำหนดรูปแบบของออบเจกต์ ใน GEE ทุกคลาสจะอยู่ภายใต้ namespace ee เช่น ee.Image, ee.Geometry

Object (Instance) เป็นสิ่งที่สร้างขึ้นมาจากการคลาสนั้นๆ ซึ่งใน GEE เมื่อเราสร้างภาพดาวเทียมหรือรูปทรงเรขาคณิต ก็จะได้เป็นออบเจกต์ของคลาสที่ต้องการ ตัวอย่างการสร้างออบเจกต์

```

var s2 = ee.ImageCollection('COPERNICUS/S2');
var roi = ee.Geometry.Rectangle([100.3, 13.6, 100.8, 14.0]);

```

Property เป็นคุณลักษณะหรือค่าคงที่ของออบเจกต์ เช่น ชื่อเบนด์, พารามิเตอร์การแสดงผล, metadata ต่างๆ

```
var feature = ee.Feature(geometry, {name: 'Chiang Mai'});
```

Method เป็นฟังก์ชันที่ผูกกับออบเจกต์ ใช้เรียกใช้งานหรือประมวลผลข้อมูล เช่น image.select(), collection.filterDate()

```

var jan2021 = s2.filterDate('2021-01-01', '2021-01-31')
  .filterBounds(roi);

var ndvi = jan2021
  .map(function (img) {
    return img.normalizedDifference(['B8', 'B4']).rename('NDVI');
  })
  .median();

```

ตัวอย่างการเรียกใช้ Vector บน Earth Engine objects

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter4.md#7-ee-objects-and-methods

```

// instance of Earth Engine objects
var geometry = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

// create a feature with properties
var feature = ee.Feature(geometry, {name: 'Chiang Mai'});
print('Feature:', feature);

// methods of Earth Engine objects
var area = feature.geometry().area();
print('Area:', area);

// Map methods
Map.centerObject(feature, 10);
Map.addLayer(feature, {color: 'red'}, 'Feature');

```

ตัวอย่างการเรียกใช้ image

```

// instance of Earth Engine objects for image
var image = ee.Image('LANDSAT/LC09/C02/T1_TOA/LC09_131047_20240103');
// methods of Earth Engine objects for image
var bandNames = image.bandNames();
print('Band names:', bandNames);
var bandCount = image.bandNames().length();
print('Band count:', bandCount);
var band4 = image.select('B4');
print('Band 4:', band4);

Map.centerObject(image, 10);
Map.addLayer(image, {bands: ['B4', 'B3', 'B2'], min: 0, max: 0.3000}, 'RGB');

```

ตัวอย่างการเรียกใช้ image collection

```
// instance of Earth Engine objects for image collection
var dataset = ee.ImageCollection('LANDSAT/LC09/C02/T1_TOA')
  .filterDate('2024-01-01', '2024-03-30') // Filter method by date;
var trueColor432 = dataset.select(['B4', 'B3', 'B2']);
var trueColor432Vis = {
  min: 0.0,
  max: 0.4,
};
Map.setCenter(98.9616, 18.7137);
Map.addLayer(trueColor432, trueColor432Vis, 'True Color (432)');
```

ตารางที่ 3 คลาสหลักใน GEE

คลาส	คำอธิบาย	ตัวอย่างการสร้างออบเจกต์
ee.Image	แทนภาพแรสเตอร์ (ดาวเทียม, ดัชนี)	<pre>var img = ee.Image('LANDSAT/LC08/C01/T1_SR/LC08_12_3032_20200715');</pre>
ee.ImageCollection	ชุดของภาพ (ee.Image) หลายภาพ	<pre>var col = ee.ImageCollection('COPERNICUS/S2')
.filterDate('2021-01-01', '2021-01-31');</pre>
ee.Geometry	รูปทรงเรขาคณิต (Point, Line, Polygon)	<pre>var pt = ee.Geometry.Point([100.5, 13.7]);</pre>
ee.Feature	เวกเตอร์ฟีเจอร์ (geometry + properties)	<pre>var f = ee.Feature(pt, {name: 'MyPoint', value: 42});</pre>
ee.FeatureCollection	ชุดของ ee.Feature หลายชิ้น	<pre>var fc = ee.FeatureCollection([f]);</pre>
ee.Number	ค่าตัวเลขเดี่ยว (scalar)	<pre>var n = ee.Number(5).add(10); // ผลลัพธ์คือ 15</pre>
ee.String	ข้อความ (text)	<pre>var s = ee.String('Hello GEE');</pre>
ee.Date	วันที่–เวลา	<pre>var d = ee.Date('2020-07-15');</pre>
ee.List	ลิสต์ของออบเจกต์ GEE ไดๆ	<pre>var lst = ee.List([1,2,3]).map(function(x){ return x.multiply(2);});</pre>
ee.Dictionary	โครงสร้าง key-value	<pre>var dict = ee.Dictionary({a:1, b:2});</pre>
ee.Array	อาร์เรย์หลายมิติ (multi-dimensional)	<pre>var arr = ee.Array([[1,2,3],[4,5,6]]);</pre>

	array)	
--	--------	--

4.9 Method Chaining

Method Chaining คือแนวทางการเขียนโค้ดที่เรียกว่า เมทอด (method) หลายๆ ตัวติดต่อกันบนขอบเจ็กต์เดียว เพื่อให้โค้ดอ่านง่ายและกระชับ โดยทุกเมทอดใน GEE (เช่น filterDate(), filterBounds(), map(), select(), median() ฯลฯ) จะคืนค่าเป็นขอบเจ็กต์ (หรือ collection) ใหม่ เมื่อเมทอดแรกคืนค่าเป็นขอบเจ็กต์ ก็สามารถเรียกเมทอดต่อไปต่อท้ายได้ทันที

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter4.md#8-method-chaining

```
// Non-chaining methods
var image = ee.Image('LANDSAT/LC09/C02/T1_TOA/LC09_131047_20240103');
var band4 = image.select('B4');
var band3 = image.select('B3');
var band2 = image.select('B2');
var rgb = band4.addBands(band3).addBands(band2);
Map.centerObject(image, 10);
Map.addLayer(rgb, {min: 0, max: 3000}, 'RGB');

// Chaining methods for image collection
var collection = ee.ImageCollection('LANDSAT/LC08/C01/T1')
  .filterDate('2020-01-01', '2020-12-31')
  .filterBounds(geometry)
  .select(['B4', 'B3', 'B2'])
  .mean();
Map.centerObject(geometry, 10);
Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Mean Image');
```

แนวทางการเขียน Chaining methods

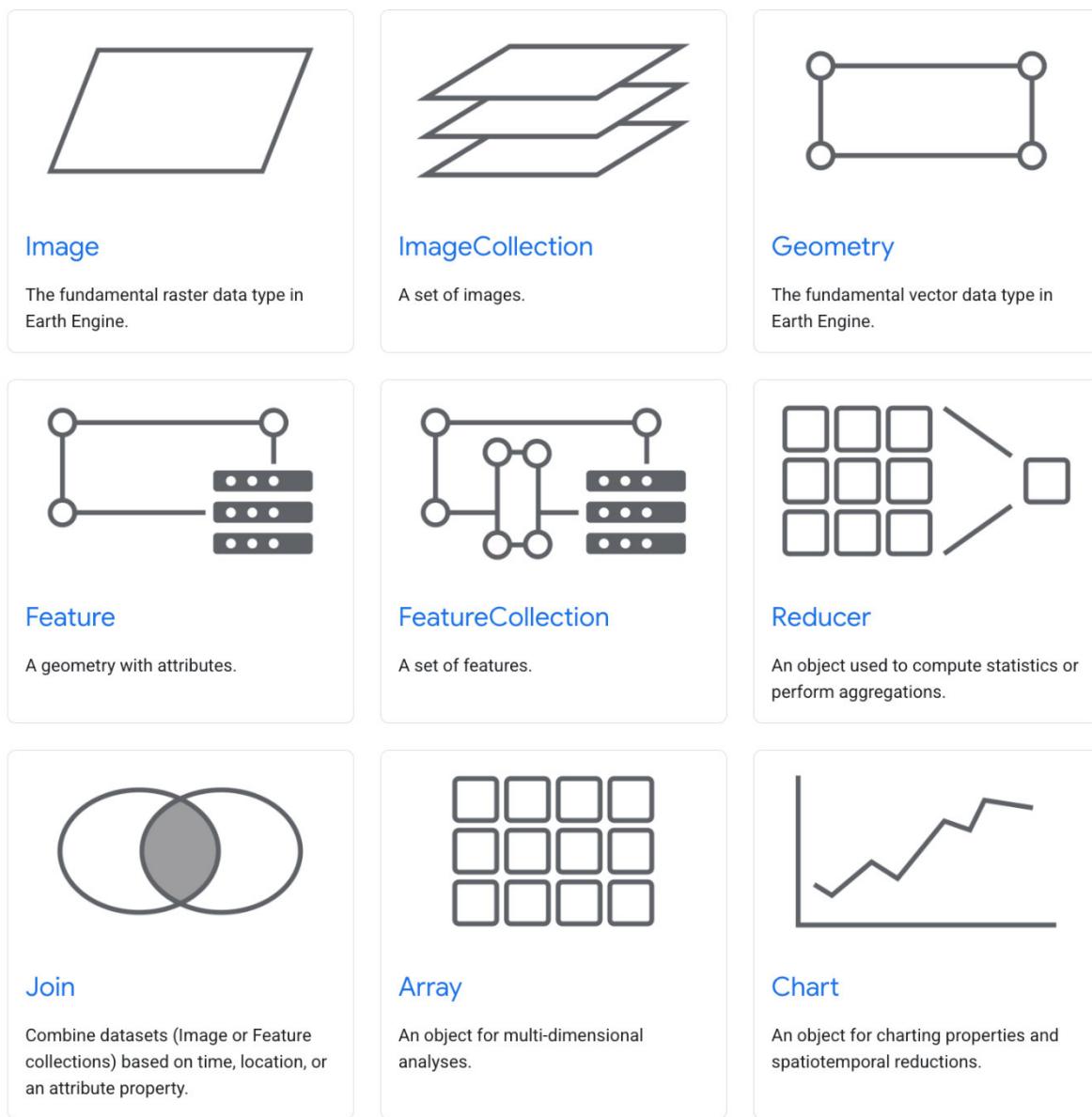
- ใช้ indentation โดยแยกแต่ละเมทอดออกจากกันและบรรทัด และยึดให้เท่ากัน
- ใส่คอมเมนต์สั้นๆ ข้างบนหรือข้างหลังแต่ละกลุ่มเมทอด
- ถ้าต้องใช้ฟังก์ชัน callback ใน .map() ให้แยกออกจากมาเขียนเป็นฟังก์ชันชื่อ เพื่อความอ่านง่าย

```
// Chaining methods with functions
function calculateNDVI(image) {
  return image.normalizedDifference(['B8', 'B4']).rename('NDVI');
}
var filtered = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2021-01-01', '2021-01-31')
  .filterBounds(geometry)
```

```
.map(calculateNDVI)
.select('NDVI')
.mean();
Map.centerObject(geometry, 10);
Map.addLayer(filtered, {min: 0, max: 1}, 'Mean NDVI');
```

บทที่ 5 GEE object

เพื่อให้เข้าใจและสามารถใช้งาน GEE ได้อย่างมีประสิทธิภาพ จะเป็นต้องทำความเข้าใจกับโครงสร้างข้อมูล และแนวคิดพื้นฐานที่สำคัญดังนี้



ภาพที่ 14 GEE object

ที่มา https://developers.google.com/earth-engine/guides/objects_methods_overview

5.1 Image

Image เป็นข้อมูลพื้นฐานที่สุดใน GEE ที่ใช้แทนภาพถ่ายดาวเทียมหรือข้อมูล Raster อีனๆ แต่ละ Image จะประกอบด้วยช่วงคลื่นหนึ่ง Bands ไปจนถึงหลาย Bands และมี Properties ที่อธิบายข้อมูลนั้นๆ Bands ช่วงคลื่นใน Image เป็นตัวแทนค่าการสะท้อนหรือการแพร่งสีของพื้นผิวโลกในช่วงความยาวคลื่นที่แตกต่างกัน เช่น ช่วงคลื่นแสงสีแดง (Red) เขียว (Green) น้ำเงิน (Blue) อินฟราเรดใกล้ (Near Infrared - NIR) หรืออินฟราเรดความร้อน (Thermal Infrared) จำนวนและชนิดของ Bands จะขึ้นอยู่กับเซนเซอร์ของดาวเทียมแต่ละดวง

Properties เป็น Metadata หรือข้อมูลที่อธิบายเกี่ยวกับ Image นั้นๆ เช่น วันที่ เวลาที่บันทึกภาพ (system: time_start) ID ของภาพ (system: id) ร้อยละการปกคลุมของเมฆ (CLOUDY_PIXEL_PERCENTAGE) หรือ ข้อมูลจำเพาะของดาวเทียม ผู้ใช้สามารถเข้าถึงและใช้ Properties เหล่านี้ในการกรองหรือวิเคราะห์ข้อมูลได้

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#1-image

```
var image = ee.Image('LANDSAT/LC09/C02/T1_L2/LC09_129050_20231220');
var band4 = image.select('SR_B4');
var band3 = image.select('SR_B3');
var band2 = image.select('SR_B2');
var rgb = band4.addBands(band3).addBands(band2);
Map.centerObject(image, 10);
Map.addLayer(rgb, {min: 8000, max: 11000}, 'RGB');
```

การสร้าง Image จาก ImageCollection

```
var bangkok = ee.Geometry.Point([100.5018, 13.7563]);
// Load Landsat 9 Collection 2 Tier 1 raw data
var landsat9 = ee.ImageCollection('LANDSAT/LC09/C02/T1_L2')
  .filterBounds(bangkok)
  .filterDate('2023-01-01', '2023-12-31')
  .sort('CLOUD_COVER')
  .first();

// Select desired spectral bands
var image = landsat9.select(['SR_B1', 'SR_B2', 'SR_B3', 'SR_B4', 'SR_B5']);

// Apply scale factors for Surface Reflectance
var opticalBands = image.select('SR_B').multiply(0.0000275).add(-0.2);

// Add image properties
var image = opticalBands
  .set({
    'system:time_start': landsat9.get('system:time_start'),
    'ACQUISITION_DATE': landsat9.date().format('YYYY-MM-dd'),
```

```

'SPACECRAFT_ID': landsat9.get('SPACECRAFT_ID'),
'CLOUD_COVER': landsat9.get('CLOUD_COVER')
});

// Print image metadata
print('Landsat 9 Image Metadata:', image);
print('Acquisition Date:', image.get('ACQUISITION_DATE'));
print('Available Band Names:', image.bandNames());

// Visualization parameters
var trueColor = {
  bands: ['SR_B4', 'SR_B3', 'SR_B2'],
  min: 0.0,
  max: 0.3,
  gamma: 1.4
};

// Center map and display
Map.centerObject(bangkok, 10);
Map.addLayer(image, trueColor, 'Landsat 9 True Color');

```

5.2 ImageCollection

ImageCollection คือกลุ่มหรือชุดของ Image ที่ถูกรวบรวมไว้ด้วยกัน โดยทั่วไปมักจะเป็นภาพจากเซนเซอร์เดียวกัน หรือมีลักษณะร่วมกันบางอย่าง เช่น ภาพถ่ายดาวเทียม Landsat 8 ทั้งหมด ภาพถ่าย Sentinel-2 ที่มีเมฆน้อยกว่า 10% ในพื้นที่ศึกษา

ImageCollection ช่วยให้สามารถจัดการและประมวลผล Image จำนวนมากพร้อมกันได้อย่างสะดวก ผู้ใช้สามารถกรอง (filter) ข้อมูลใน ImageCollection ตามช่วงเวลา ขอบเขตพื้นที่ หรือคุณสมบัติอื่นๆ สามารถจัดเรียง (sort) และนำฟังก์ชันต่างๆ มาประยุกต์ใช้กับทุก Image ใน Collection ได้ เช่น การสร้างภาพโมเสค (mosaic) หรือการคำนวนค่าเฉลี่ยตามช่วงเวลา

source code:
https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#2-image-collection

```

var geometry = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
var collection = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2021-01-01', '2021-01-31')
  .filterBounds(geometry)
  .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
  .select(['B4', 'B3', 'B2'])
  .median();
Map.centerObject(geometry, 10);

```

```
Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Median Image');
```

5.3 Geometry

Geometry ใน Google Earth Engine (GEE) คือออบเจ็กต์ที่ใช้แทนรูปทรงเรขาคณิตต่างๆ เพื่อกำหนดพื้นที่ วิเคราะห์และทำงานกับข้อมูลเชิงพื้นที่ โดยอยู่ภายใต้คลาส ee.Geometry ซึ่งรองรับประเภทหลักดังนี้

- Point: จุดเดียวบนแผนที่
- MultiPoint: หลายจุดรวมกัน
- LineString: เส้นเชื่อมระหว่างจุดเรียงกัน
- MultiLineString: หลายเส้นรวมกัน
- Polygon: พื้นที่ปิด (ขอบเขต)
- MultiPolygon: หลาย Polygon รวมกัน
- Rectangle: สี่เหลี่ยม กำหนดด้วยมุมซ้ายล่างและขวาบน

Source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#3-geometry

```
// Geometry object
var point = ee.Geometry.Point([98.9171009716561, 18.815619476862654]);
var line = ee.Geometry.LineString([[98.9171009716561, 18.815619476862654],
[99.0873890575936, 18.68557890893041]]);
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
  [98.9171009716561, 18.68557890893041],
  [99.0873890575936, 18.68557890893041],
  [99.0873890575936, 18.815619476862654]]]);
var buffer = point.buffer(1000); // Buffer of 1000 meters
var centroid = polygon.centroid();
var area = polygon.area(); // Area of the polygon
print('Point:', point);
print('Line:', line);
print('Polygon:', polygon);
print('Buffer:', buffer);
print('Centroid:', centroid);
print('Area:', area);
// Map visualization
Map.centerObject(polygon, 10);
Map.addLayer(point, {color: 'yellow'}, 'Point');
Map.addLayer(line, {color: 'orange'}, 'Line');
Map.addLayer(polygon, {color: 'red'}, 'Polygon');
Map.addLayer(buffer, {color: 'blue'}, 'Buffer');
Map.addLayer(centroid, {color: 'green'}, 'Centroid');
```

5.4 Feature

Feature ใน GEE ใช้แทน Object หรือวัตถุต่างๆ ที่เป็นตัวแทนข้อมูลทางภูมิศาสตร์ ซึ่งประกอบด้วยสองส่วน หลักคือ Geometry และ Properties

- Geometry คือ ข้อมูลที่อธิบายรูปร่างและตำแหน่งของวัตถุบนพื้นโลก สามารถเป็นได้ทั้ง Point (จุด เช่น ตำแหน่งโรงเรียน) LineString (เส้น เช่น ถนน แม่น้ำ) หรือ Polygon (รูปปิด เช่น ขอบเขต จังหวัด พื้นที่ทะเลสาบ)
- Properties เป็นข้อมูลอธิบายลักษณะของ `Feature` นั้นๆ เช่น ชื่อสถานที่, ประเภท, ความยาว, พื้นที่ หรือข้อมูลอื่นๆ ที่เกี่ยวข้อง ในรูปแบบของ key-value pairs

ตัวอย่าง ตำแหน่งของชุมชน จะประกอบด้วย ตัวแทนที่เป็นจุด (มี Geometry เป็น Point) และคุณสมบัติ (Properties) เช่น ชื่อชุมชน หรือ ขอบเขตของชุมชนในจังหวัดเชียงใหม่จะประกอบด้วย ตัวแทนที่เป็นพื้นที่รูปปิด (มี Geometry เป็น Polygon) พร้อมด้วยคุณสมบัติ เช่น จำนวนประชากร พื้นที่ทั้งหมด

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#4-feature

```
//Feature object
var point = ee.Geometry.Point([98.9171009716561, 18.815619476862654]);
var feature = ee.Feature(point, {name: 'Chiang Mai', population: 1000000});
print('Feature:', feature);

// Polygon feature
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
var feature = ee.Feature(polygon, {name: 'Chiang Mai', population: 1000000});
print('Feature:', feature);
// add to map
Map.centerObject(feature, 10);
Map.addLayer(feature, {color: 'red'}, 'Feature');
```

5.5 FeatureCollection

FeatureCollection คือกลุ่มหรือชุดของ Feature ที่ถูกรวบรวมไว้ด้วยกัน คล้ายกับ Shapefile หรือ GeoJSON ที่เก็บข้อมูลจากเตอร์หลายๆ ชิ้น ด้านการจัดการและการประมวลผลความสามารถ (filter)

ข้อมูลใน FeatureCollection ตามคุณสมบัติหรือขอบเขตพื้นที่ และนำไปใช้ในการวิเคราะห์ร่วมกับข้อมูล Raster ได้ เช่น การนำข้อมูลนี้มาทำการตัด (clip) ภาพถ่ายดาวเทียม หรือการคำนวนค่าเฉลี่ยของดัชนีพิชพรผลภายในแต่ละแปลงเกษตร (FeatureCollection) ตัวอย่าง ชุดข้อมูลขอบเขตการปกครองระดับจังหวัดทั้งหมดของประเทศไทย ชุดข้อมูลตำแหน่งหมู่บ้านทั้งหมดในภาคเหนือ หรือชุดข้อมูลเส้นทางถนนทั้งหมดในกรุงเทพมหานคร

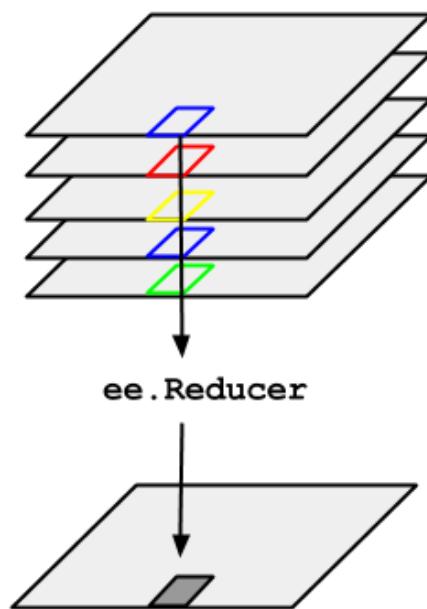
source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#5-feature-collection

```
// Feature Collection object
var point1 = ee.Geometry.Point([98.9171009716561, 18.815619476862654]);
var point2 = ee.Geometry.Point([99.0873890575936, 18.68557890893041]);
var feature1 = ee.Feature(point1, {name: 'Chiang Mai', population: 1000000});
var feature2 = ee.Feature(point2, {name: 'Sarapee', population: 8000000});
var featureCollection = ee.FeatureCollection([feature1, feature2]);
print('Feature Collection:', featureCollection);
// add to map
Map.centerObject(featureCollection, 10);
Map.addLayer(featureCollection, {color: 'red'}, 'Feature Collection');
```

5.6 Reducer

Reducer เป็นฟังก์ชันที่ใช้ในการสรุปหรือรวม (aggregate) ข้อมูลจาก Image ImageCollection หรือ FeatureCollection ให้เป็นค่าเดียวหรือชุดค่าที่มีขนาดเล็กลง ทำให้ง่ายต่อการวิเคราะห์และแสดงผล



การทำงานของ Reducer จะรับข้อมูลนำเข้า (input) ที่เป็นชุดของค่า (เช่น ค่า pixel ทั้งหมดใน Image หรือ

ในแต่ละ Band ค่าคุณสมบัติของ Feature ใน FeatureCollection และจำนวนค่าสรุปอุปกรณ์ตามที่กำหนด การใช้งาน Reducer ใน Google Earth Engine สามารถแบ่งตามมิติหลักได้ 3 กลุ่ม คือ เวลา (Time), พื้นที่ (Space) และ แบนด์ (Band) ซึ่งแต่ละมิติจะใช้รีดิวเซอร์ต่างกันตามลักษณะข้อมูล ดังนี้

- **Time Reducers:** สรุปค่าข้ามเวลา → ใช้บน ImageCollection ด้วย .reduce(...) หรือแยกกลุ่มตามเวลา
- **Spatial Reducers:** สรุปค่าตามพื้นที่ → ใช้ reduceRegion (ROI) และ reduceNeighborhood (เครื่องเรียนรู้รอบพิกเซล)
- **Band Reducers:** สรุปค่าตามแบนด์ → ใช้ reduceRegion บนหลายๆ แบนด์ และ reduce เพื่อรวมค่าแบนด์

5.6.1 Time Reducers (สรุปตามเวลา)

1) Per-pixel time series

ใช้ Reducers บนชุดภาพ (ImageCollection) เพื่อสรุปค่าสำหรับแต่ละพิกเซลข้ามช่วงเวลา ตัวอย่างด้านล่าง เราจะได้ผลลัพธ์คือ ee.Image ใหม่ ที่แต่ละพิกเซลเก็บค่าเฉลี่ยข้ามปี

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#6-reducer

```
// Per-pixel time series
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2021-01-01', '2021-12-31')
  .filterBounds(polygon);

// Compute per-pixel mean across time
var meanTime = s2.reduce(ee.Reducer.mean());

Map.addLayer(meanTime, {bands: ['B4_mean', 'B3_mean', 'B2_mean'], min:0, max:3000},
  'Mean per Pixel over Time');
```

2) Grouped by time

เป็นจัดกลุ่มตามช่วงเวลา ใช้สร้าง histogram ของจำนวนภาพในแต่ละช่วงเวลา เช่น เดือนหรือปี ตัวอย่าง เช่น นับจำนวนภาพ Sentinel-2 แต่ละเดือนในปี 2021

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#6-reducer

```
// Grouped by time
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2021-01-01', '2021-12-31')
  .filterBounds(polygon);
var months = ee.List.sequence(1, 12);

var monthlyCount = months.map(function(m) {
  var filtered = s2.filter(ee.Filter.calendarRange(m, m, 'month'));
  return filtered.size();
});

print('Images per month (2021):', monthlyCount);

// add histogram chart
var chart = ui.Chart.array.values(monthlyCount, 0, months)
  .setChartType('ColumnChart')
  .setOptions({
    title: 'Monthly Image Count (2021)',
    hAxis: {title: 'Month'},
    vAxis: {title: 'Image Count'},
    legend: {position: 'none'}
  });
print(chart);
```

5.6.2 Spatial Reducers (สรุปเชิงพื้นที่)

1) Regional statistics (reduceRegion)

เป็นการสรุปค่าสถิติในพื้นที่ (ROI) โดยใช้ เช่น mean, sum, count ตัวอย่างด้านล่าง ได้ผลลัพธ์เป็น dictionary ที่มีค่าคงแหนณเฉลี่ยและค่าสูงสุดของทุกแบบนัด ในพื้นที่ Polygon

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#image-statistics-reduceregion

```
// Regional statistics (reduceRegion)
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]];

var s2 = ee.ImageCollection('COPERNICUS/S2')
```

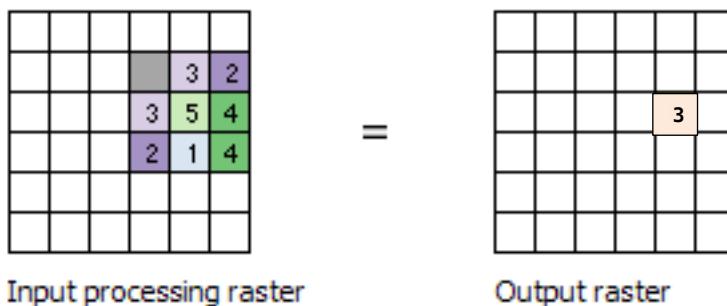
```

    .filterDate('2021-01-01', '2021-03-31')
    .filterBounds(polygon);
var meanTime = s2.reduce(ee.Reducer.mean());
var stats = meanTime.reduceRegion({
  reducer: ee.Reducer.mean().combine({
    reducer2: ee.Reducer.max(),
    sharedInputs: true
  }),
  geometry: polygon,
  scale: 30
});
print('Mean & Max over polygon:', stats);

```

2) Neighborhood / Focal operations (reduceNeighborhood)

เป็นการสรุปค่าสำหรับเครื่องในครอบแต่ละพิกเซล เช่น focal mean, focal max ตัวอย่างด้านล่างเป็นการทำ focal mean 3x3 เพื่อกำกับการกรอง noise หรือเน้นโครงสร้างหรือรูปร่างของวัตถุในภาพ



ที่มา <https://desktop.arcgis.com/en/arcmap/latest/tools/spatial-analyst-toolbox/how-focal-statistics-works.htm>

source code:

```

https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter5.md#neighborhood--focal-operations-reduceneighborhood
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

```

```

var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2021-01-01', '2021-12-31')
  .filterBounds(polygon);
var meanTime = s2.reduce(ee.Reducer.mean());
// Neighborhood / Focal operations (reduceNeighborhood)
var focalMean = meanTime.reduceNeighborhood({
  reducer: ee.Reducer.mean(),
  kernel: ee.Kernel.square({radius: 1})
}

```

```
});
Map.addLayer(focalMean, {min:0, max:3000}, '3x3 Focal Mean');
```

5.6.3 Band Reducers

1) Per-band summary (reduceRegion on multiband) เป็นการสรุปสถิติค่าเฉลี่ยแยกตามแบบด้านภาพเดียวกัน

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#per-band-summary-reduceregion-on-multiband

```
// Per-band summary (reduceRegion on multiband)
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2021-01-01','2021-03-31')
  .filterBounds(polygon);
var composite = s2.median();
var bandStats = composite.reduceRegion({
  reducer: ee.Reducer.mean(),
  geometry: polygon,
  scale: 100
});
print('Mean per band:', bandStats);
```

2) Across-band reduction (reduce) เป็นการสรุปค่าสถิติข้ามแบบดั้งเดิม เช่น sum, variance ตัวอย่างด้านล่างเป็นการรวมค่าทุกแบบด้วยวิธีเดียว

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter5.md#across-band-summary-reduceregion-on-multiband

```
// Across-band reduction (reduce)
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]];

var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2021-01-01','2021-12-31')
  .filterBounds(polygon);
var composite = s2.median();
var bandSum = composite.reduce(ee.Reducer.sum());
Map.addLayer(bandSum, {min:9000, max:15000}, 'Sum across Bands');
```

5.7 Join

Join เป็นกระบวนการที่ใช้ในการรวมหรือเชื่อมโยงข้อมูลจาก ImageCollection หรือ FeatureCollection สองชุดหรือมากกว่าเข้าด้วยกัน โดยอาศัยเงื่อนไขบางอย่างที่กำหนดความสัมพันธ์ระหว่างข้อมูลเหล่านั้น เช่น ใช้เวลา (เช่น การจับคู่ภาพถ่ายดาวเทียมที่ถ่ายในวันเดียวกัน หรือใกล้เคียงกัน) หรือความสัมพันธ์เชิงพื้นที่ (เช่น การจับคู่ข้อมูลที่อยู่ในตำแหน่งเดียวกันหรือซ่อนทับกัน) หรือการจับคู่ตาม Property ที่มีค่าเหมือนกัน

source code:

```
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter5.md#7-join
// Define point and polygon collections
var points = ee.FeatureCollection([
  ee.Feature(ee.Geometry.Point([100.5,13.7]), {pid:1}),
  ee.Feature(ee.Geometry.Point([100.55,13.75]), {pid:2})
]);
var polygons = ee.FeatureCollection([
  ee.Feature(ee.Geometry.Rectangle([100.4,13.6,100.6,13.8]), {zone: 'A'}),
  ee.Feature(ee.Geometry.Rectangle([100.45,13.65,100.65,13.85]), {zone: 'B'})
]);

// Create a spatial filter: point within polygon
var spatialFilter = ee.Filter.contains({
  leftField: '.geo',      // polygon geometry
  rightField: '.geo'       // point geometry
});

// Perform an inner spatial join
var spatialJoin = ee.Join.inner();
var spatialJoined = spatialJoin.apply(polygons, points, spatialFilter);

// Attach point property 'pid' to each polygon
var result = spatialJoined.map(function(f) {
  var poly = ee.Feature(f.get('primary'));
  var pt   = ee.Feature(f.get('secondary'));
  return poly.set('point_id', pt.get('pid'));
});

// Display on the map
Map.centerObject(polygons, 10);

// Polygons in blue
Map.addLayer(polygons, {color: 'blue'}, 'Polygons');

// Points in black
Map.addLayer(points, {color: 'black'}, 'Points');

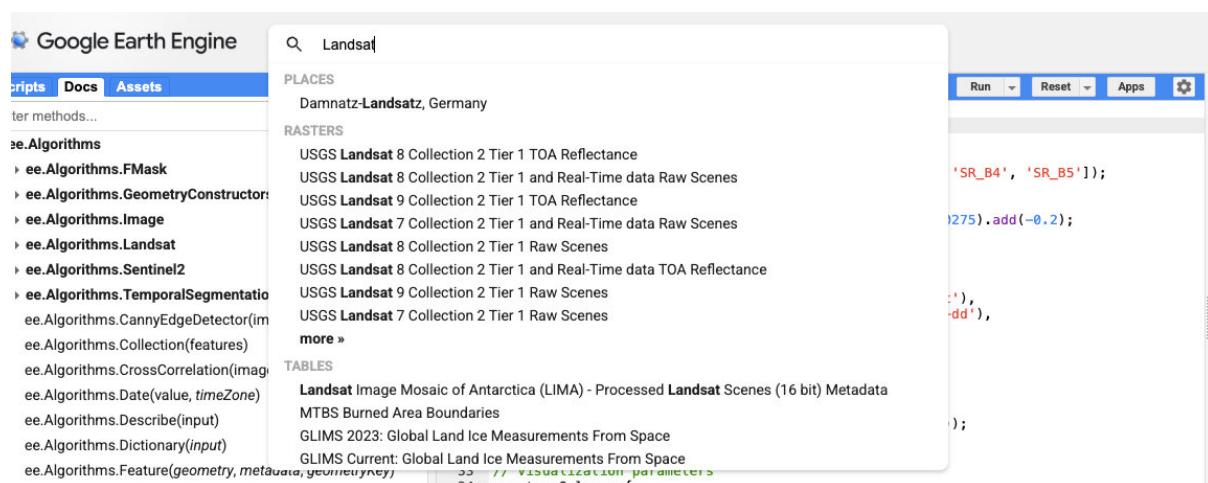
// Spatially-joined polygons with attached pid in purple
Map.addLayer(result, {color: 'purple'}, 'Spatial Join');
```

บทที่ 6 การทำงานกับข้อมูลภาพดาวเทียม

หัวใจสำคัญของการใช้งาน GEE คือการเข้าถึงและวิเคราะห์ข้อมูลภาพถ่ายดาวเทียมจำนวนมากที่มีอยู่ในคลังข้อมูล GEE Code Editor มีเครื่องมือและฟังก์ชันที่ช่วยให้การค้นหา เรียกใช้ และแสดงผลข้อมูลเหล่านี้ทำได้โดยง่าย

6.1 การค้นหาข้อมูลจากคลังข้อมูล GEE

ใช้ช่องค้นหา (Search bar) เป็นวิธีที่ง่ายที่สุดในการค้นหาชุดข้อมูลคือการใช้ช่องค้นหาที่อยู่ด้านบนของ Code Editor พิมพ์ชื่อดาวเทียม (เช่น "Landsat 8", "Sentinel-2") หรือประเภทข้อมูลที่สนใจ (เช่น "DEM", "precipitation") ระบบจะแสดงรายการชุดข้อมูลที่เกี่ยวข้องขึ้นมา



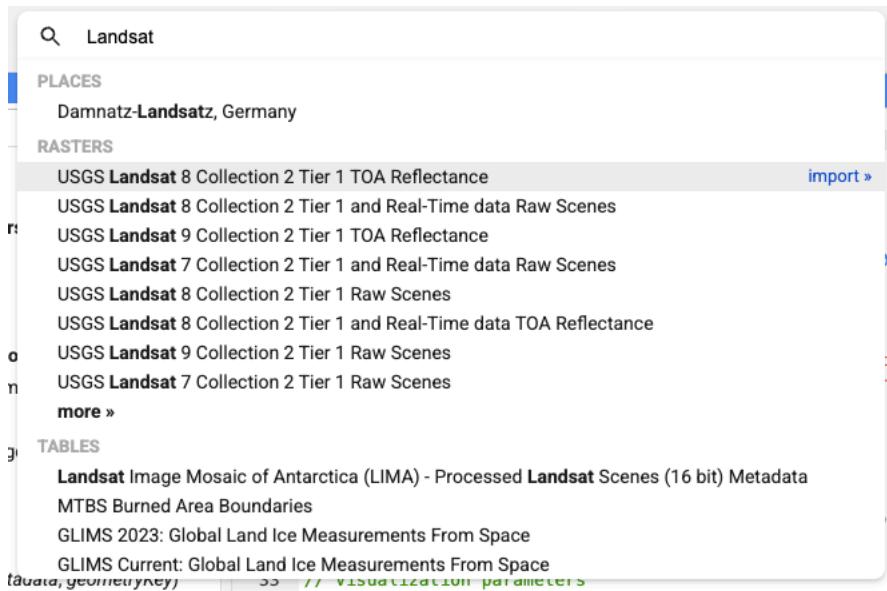
The screenshot shows the Google Earth Engine Code Editor interface. On the left, there's a sidebar with tabs for Scripts, Docs, and Assets. Below that is a list of methods under ee.Algorithms. In the center, there's a search bar with the text 'Landsat'. Below the search bar, there are two sections: PLACES and RASTERS. The PLACES section shows 'Darmstadt-Landsat, Germany'. The RASTERS section lists various Landsat datasets from USGS and MTBS. On the right, the code editor shows a snippet of JavaScript code:

```
'SR_B4', 'SR_B5']);
1275).add(-0.2);
:'),
-dd'),
```

Below the code editor, there are tabs for Run, Reset, Apps, and Settings.

เมื่อคลิกที่ชุดข้อมูลจากผลการค้นหา จะมีหน้าต่างแสดงรายละเอียดของชุดข้อมูลนั้นๆ เช่น คำอธิบาย ช่วงเวลาที่มีข้อมูล Bands ที่มี ตัวอย่างโค้ดในการเรียกใช้ และที่สำคัญคือ ImageCollection ID หรือ Image ID ซึ่งเป็นรหัสที่ใช้อ้างอิงถึงชุดข้อมูลนั้นในสคริปต์

ใช้ปุ่ม "Import" ในหน้าต่างรายละเอียดชุดข้อมูล จะมีปุ่ม "Import" เมื่อคลิกปุ่มนี้ GEE จะเพิ่มบรรทัดโค้ดสำหรับเรียกใช้ชุดข้อมูลนั้นเข้ามาในสคริปต์ของผู้ใช้โดยอัตโนมัติ โดยมักจะกำหนดเป็นตัวแปรชื่อ imageCollection หรือ image ซึ่งผู้ใช้สามารถเปลี่ยนชื่อได้



ตัวอย่างการ Import Landsat 8 Collection 2 Tier 1 Surface Reflectance GEE จะสร้างบรรทัดนี้ให้เมื่อ กด Import

```
var imageCollection = ee.ImageCollection("LANDSAT/LC08/C02/T1_TOA");
var collection = ee.ImageCollection('COPERNICUS/S2')
```

6.2 การเรียกค่า properties

การสร้าง Object ImageCollection เป็นการเรียกใช้ชุดข้อมูลภาพถ่ายดาวเทียมส่วนใหญ่จะเริ่มต้นด้วยการ สร้างอ็อบเจกต์ ee.ImageCollection โดยระบุ ID ของชุดข้อมูลนั้นๆ

```
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

var collection = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-01-01', '2025-03-31')
  .filterBounds(polygon)

Map.centerObject(polygon, 10);
Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Image Collection');
```

หลังจากเรียกใช้ ImageCollection แล้ว สามารถใช้คำสั่ง print() เพื่อดูรายละเอียดเบื้องต้นของ Collection ใน Console ได้ เช่น จำนวนภาพทั้งหมด (อาจต้องใช้ .size() ซึ่งจะคำนวณบน Server) Properties ของ Collection และตัวอย่าง Properties ของ Image แรกใน Collection

source code:

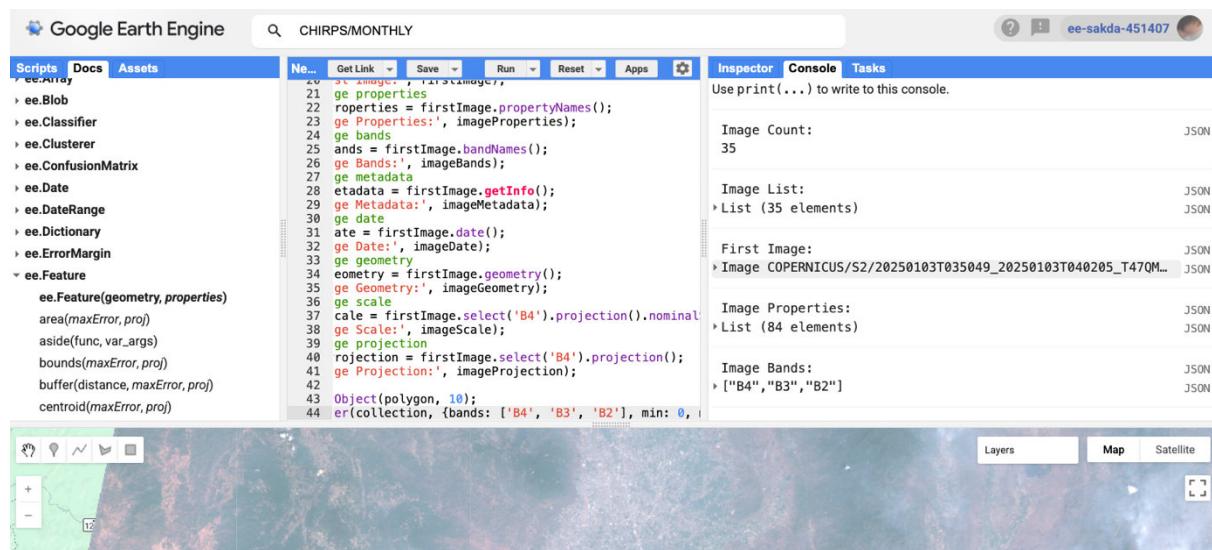
https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#1-images-properties

```
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

var collection = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-01-01', '2025-03-31')
  .filterBounds(polygon)
  .select(['B4', 'B3', 'B2'])

// get image count
var imageCount = collection.size();
print('Image Count:', imageCount);
// get image list
var imageList = collection.toList(imageCount);
print('Image List:', imageList);
// get first image
var firstImage = ee.Image(imageList.get(0));
print('First Image:', firstImage);
// get image properties
var imageProperties = firstImage.propertyNames();
print('Image Properties:', imageProperties);
// get image bands
var imageBands = firstImage.bandNames();
print('Image Bands:', imageBands);
// get image metadata
var imageMetadata = firstImage.getInfo();
print('Image Metadata:', imageMetadata);
// get image date
var imageDate = firstImage.date();
print('Image Date:', imageDate);
// get image geometry
var imageGeometry = firstImage.geometry();
print('Image Geometry:', imageGeometry);
// get image scale
var imageScale = firstImage.select('B4').projection().nominalScale();
print('Image Scale:', imageScale);
// get image projection
var imageProjection = firstImage.select('B4').projection();
print('Image Projection:', imageProjection);
```

ผลลัพธ์การทำงานของโค้ด



6.3 การแสดงผลภาพบนแผนที่

การแสดงผลภาพบนแผนที่ จะใช้ฟังก์ชัน Map.addLayer() เป็นฟังก์ชันหลักที่ใช้ในการนำ ee.Image, ee.ImageCollection (โดยทั่วไปจะแสดงภาพแรกหรือภาพที่ผ่านการ reduce เช่น median mosaic), ee.Feature, หรือ ee.FeatureCollection มาแสดงผลบนแผนที่ใน Code Editor พารามิเตอร์ของฟังก์ชัน Map.addLayer() ประกอบด้วย

- eeObject: อีอบเจกต์ GEE ที่ต้องการแสดงผล (เช่น ee.Image, ee.FeatureCollection)
- visParams (Visualization Parameters): อีอบเจกต์ JavaScript ที่กำหนดค่าการแสดงผล เช่น Bands ที่จะใช้, ค่า min/max ของสี, palette สี (สำหรับภาพแบบ single-band)
- name (Optional): ชื่อของ Layer ที่จะปรากฏในส่วนจัดการ Layer บนแผนที่
- shown (Optional): Boolean (true/false) กำหนดว่าจะให้ Layer แสดงผลทันทีหรือไม่ (default คือ true)
- opacity (Optional): ค่าความโปร่งใสของ Layer (0.0 ถึง 1.0)

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#2-map-object

```

var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
  
```

```

var collection = ee.ImageCollection('COPERNICUS/S2')
  
```

```

    .filterDate('2025-01-01', '2025-03-31')
    .filterBounds(polygon)
Map.centerObject(collection, 8);
Map.addLayer(
  collection, // eeObject
  {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, // visParams
  'Image Collection', // name
  true, // shown (hidden by default)
  0.8 // opacity (80% transparent)
);

```

ผลลัพธ์การทำงานของโค้ด

The screenshot shows the Google Earth Engine (GEE) web interface. In the top navigation bar, it says "Google Earth Engine" and "CHIRPS/MONTHLY". Below the navigation is a toolbar with "Script", "Docs", and "Assets" buttons. The "Script" tab is active, showing a "New Script" area with the provided JavaScript code. The code defines a polygon, filters an image collection for the specified date range and bands, centers the map on the collection, adds a layer, and sets visualization parameters. The main workspace below the toolbar shows a map of northern Thailand, specifically the Chiang Mai region, with a polygon drawn over it. The map includes labels for cities like Chiang Mai, Lampang, and Nan, and various rivers and roads. On the right side of the interface, there are panels for "Inspector", "Console", and "Tasks". The "Console" panel contains the message "Use print(...) to write to this console.".

การแสดงผลภาพถ่ายดาวเทียมในแน่นอนใจและสื่อความหมายเป็นสิ่งสำคัญ GEE ให้เราควบคุมพารามิเตอร์การแสดงผล (Visualization Parameters) ได้อย่างละเอียดผ่านอ็อบเจกต์ JavaScript ที่ส่งเข้าไปใน Map.addLayer() โดยการกำหนดสี (Palettes) และการผสมสี (Band Combinations)

6.3.1 ภาพสีผสม (RGB Composite) ใช้สำหรับภาพหลายช่วงคลื่น (Multispectral Images)

เราจะสามารถสร้างภาพสีผสมโดยการกำหนดให้ Band ใดแสดงเป็นสีแดง (Red) เขียว (Green) และน้ำเงิน (Blue) มีรูปแบบ ดังนี้

```

var rgbVis = {
  bands: ['B4', 'B3', 'B2'], // Use red, green, blue bands
  min: 0, // Map pixel values from 0
  max: 3000, // to 3000
  gamma: 1.1 // Apply slight gamma correction
};

```

โดยที่

- Bands: เป็น Property ใน `visParams` ที่เป็น List ของชื่อ Bands 3 Bands ตามลำดับ [Red, Green, Blue]
- min และ max: สามารถกำหนดเป็น List ของค่าต่ำสุดและสูงสุดสำหรับแต่ละ Band ที่เลือก หรือ กำหนดเป็นค่าเดียวที่จะใช้กับทุก Band
- gamma: เป็นพารามิเตอร์ที่ใช้ปรับแก้ Gamma Correction ของภาพ ทำให้ภาพสว่างขึ้นหรือมีดลง โดยไม่เชิงเส้น ค่า Default คือ 1.0 (gamma > 1.0: ภาพจะมีดลง ขณะที่ gamma < 1.0: ภาพจะสว่างขึ้น)

ตัวอย่างการแสดงผลภาพสีจริง (True Color Composite) สำหรับ Landsat 8 SR

```
var trueColorVis = {
  bands: ['SR_B4', 'SR_B3', 'SR_B2'], // Red, Green, Blue
  min: 0.0,
  max: 0.3,
  gamma: 1.1
};
```

ตัวอย่างการแสดงผลภาพสีเท็จ (False Color Composite) - เน้นพืชพรรณ

```
var falseColorVis_Veg = {
  bands: ['SR_B5', 'SR_B4', 'SR_B3'], // NIR, Red, Green
  min: 0.0,
  max: 0.4,
  gamma: 1.4
};
```

6.3.2 ภาพแบบ Band เดียว (Single-Band Image) กับ Palette สี

สำหรับภาพที่มี Band เดียว (เช่น DEM, NDVI, หรือ Band เดียวๆ) เราสามารถใช้ Palette สีเพื่อกำหนดว่า ค่า Pixel แต่ละดวงจะแสดงเป็นสีอะไร มีรูปแบบ ดังนี้

```
var demVis = {
  min: 0, // lowest elevation (meters)
  max: 3000, // highest elevation (meters)
  palette: [
    '0000ff', // deep water (if below 0)
    '00ffff', // sea level
    '00ff00', // lowlands
    'ffff00', // mid elevations
    'ff7f00', // high elevations
    'ffffff' // peaks
  ]
};
```

โดยที่

- palette เป็น Property ใน visParams ที่เป็น List ของสี (Hex color codes)
- ค่า min และ max เป็นพารามิเตอร์ที่สำคัญที่สุดในการควบคุมการยืดสี (Contrast Stretching) ค่า Pixel ที่ต่ำกว่า min จะแสดงเป็นสีแรกใน Palette (หรือสีดำใน RGB) และค่าที่สูงกว่า max จะแสดงเป็นสีสุดท้าย (หรือสีขาวใน RGB) การเลือกค่า min และ max ที่เหมาะสมจะช่วยให้เห็นรายละเอียดในภาพได้ชัดเจนขึ้น

ตัวอย่างการแสดงผล NDVI

```
var ndviVis = {
  min: -0.5,
  max: 0.8,
  palette: [
    '#d73027',
    '#fc8d59',
    '#fee08b',
    '#ffffbf',
    '#d9ef8b',
    '#91cf60',
    '#1a9850'
  ];
};
```

ตัวอย่างการแสดงผล DEM (Digital Elevation Model)

```
var demVis = {
  min: 0,
  max: 4000,
  palette: [
    '006633',
    'E5FFCC',
    '662A00',
    'D8D8D8',
    'F5F5F5'
  ]
};
```

ตัวอย่างด้านล่างเป็นการแสดงผลทั้งแบบภาพสีผสม (RGB Composite) ใช้สำหรับภาพหลายช่วงคลื่น (Multispectral Images) และ ภาพแบบ Band เดียว (Single-Band Image) กับ Palette สี

Source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#3-visulization

```
var polygon = ee.Geometry.Polygon(
  [[[-98.9171009716561, 18.815619476862654],
    [-98.9171009716561, 18.68557890893041],
```

```

[99.0873890575936, 18.68557890893041],
[99.0873890575936, 18.815619476862654]]);

var collection = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-01-01', '2025-03-31')
  .filterBounds(polygon)

var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017')
  .filter(ee.Filter.eq('country_na', 'Thailand'));

var dem = ee.Image('USGS/SRTMGL1_003');

var rgbVis = {
  bands: ['B4', 'B3', 'B2'], // Use red, green, blue bands
  min: 0, // Map pixel values from 0
  max: 3000, // to 3000
  gamma: 1.1 // Apply slight gamma correction
};

var demVis = {
  min: 0, // lowest elevation (meters)
  max: 3000, // highest elevation (meters)
  palette: [
    '0000ff', // deep water (if below 0)
    '00ffff', // sea level
    '00ff00', // lowlands
    'ffff00', // mid elevations
    'ff7f00', // high elevations
    'ffffff' // peaks
  ]
};

var countryStyle = {
  color: 'FF0000', // Red outline
  fillColor: 'FF000022', // Translucent red fill
  width: 1 // 1-pixel wide border
};

Map.centerObject(collection, 8); // Zoom level 8

// Add the RGB image layer
Map.addLayer(
  collection, // eeObject
  rgbVis, // visParams
  'Sentinel-2 RGB', // name
  true, // shown
  0.8 // opacity
);

// Add the DEM layer
Map.addLayer(

```

```

    dem,                                // eeObject
    demVis,                             // visParams
    'SRTM DEM',                         // name
    false,                               // shown (hidden by default)
    0.5                                 // opacity (50% transparent)
);

// Add the country borders layer
Map.addLayer(
  countries,                          // eeObject
  countryStyle,                      // visParams
  'Country Borders',                // name
  false,                               // shown (hidden by default)
  1.0                                 // opacity (fully opaque)
);

```

ผลลัพธ์การทำงานของโค้ด

The screenshot shows the Google Earth Engine interface. In the top navigation bar, it says "Google Earth Engine" and "CHIRPS/MONTHLY". Below the navigation bar, there are tabs for "Scripts", "Docs", and "Assets", with "Scripts" selected. The main area is titled "New Script *". The script code is identical to the one above, showing the addition of a "countries" layer to the map. To the right of the code editor is the "Inspector", "Console", and "Tasks" panel, which is currently empty. Below the code editor is the map view, which displays a satellite image of a coastal region with a semi-transparent gray polygon overlaid on the land area. The map includes standard controls for zooming and panning.

6.4 การกรองข้อมูล (filter)

ImageCollection ที่ได้จากการคลังข้อมูล GEE มักจะมีภาพจำนวนมากครอบคลุมพื้นที่กว้างและช่วงเวลา ยาวนาน การกรอง (Filtering) เป็นขั้นตอนสำคัญในการเลือกเฉพาะภาพที่ตรงกับความต้องการในการ วิเคราะห์ของผู้ใช้ GEE มีฟังก์ชันการกรองที่หลากหลาย

6.4.1 การกรองตามช่วงเวลา (filterDate)

ฟังก์ชัน filterDate() ใช้สำหรับกรอง ImageCollection ให้เหลือเฉพาะภาพที่บันทึกภายในช่วงวันที่และเวลา ที่กำหนด พารามิเตอร์ประกอบด้วย

- startDate: วันที่เริ่มต้น (String ในรูปแบบ 'YYYY-MM-DD' หรือ ee.Date)
- endDate (Optional): วันที่สิ้นสุด (ถ้าไม่ระบุ จะหมายถึงภาพทั้งหมดหลังจาก startDate)

```
// Filter by date
var collection = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-01-01', '2025-03-31') // Filter method by date
Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Image Collection');
```

หมายเหตุ endDate จะเป็นแบบ exclusive คือไม่รวมภาพที่ตรงกับ endDate พอดี หากต้องการรวมให้ระบุวันถัดไป เช่น กรองถึงสิ้นปี 2023 ให้ใช้ '2024-01-01' เป็น endDate

6.4.2 การกรองตามขอบเขตพื้นที่ (filterBounds)

ฟังก์ชัน filterBounds() ใช้สำหรับกรอง ImageCollection ให้เหลือเฉพาะภาพที่ซ้อนทับ (intersect) กับขอบเขตพื้นที่ (Geometry) ที่กำหนด พารามิเตอร์ประกอบด้วย

- geometry: ee.Geometry เป็น object ที่ใช้เป็นขอบเขตในการกรอง (เช่น ee.Geometry.Point, ee.Geometry.Polygon, หรือ Feature, FeatureCollection)
- การสร้าง Geometry สามารถทำได้หลายแนวทาง ดังนี้
 - วาดบนแผนที่ ใน Code Editor มีเครื่องมือให้วาด Point, Line, หรือ Polygon บนแผนที่โดยตรง Geometry ที่วาดจะถูกสร้างเป็นตัวแปรชื่อ geometry (หรือชื่ออื่นที่กำหนด) โดยอัตโนมัติ
 - สร้างจากพิกัด เช่น `var point = ee.Geometry.Point([longitude, latitude]);`
 - สร้างจาก FeatureCollection เช่น ขอบเขตจังหวัดที่ Import มาจาก Asset

source code:

```
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter6.md#4-filter
// Filter by bounds
var geometry = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
var collection = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-01-01', '2025-03-31') // Filter method by date
  .filterBounds(geometry) // Filter method by bounds
Map.centerObject(geometry, 10);
Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Image Collection');
```

6.4.3 การกรองตามคุณสมบัติอื่นๆ (เช่น เปรอเซ็นต์เมջ filterMetadata หรือ ee.Filter)

พังก์ชัน ee.Filter ใช้สำหรับกรอง ImageCollection โดยอิงจากค่าของ Property (Metadata) ของแต่ละ Image ตัวอย่างการกรองภาพที่มีเมฆน้อย (Sentinel-2 มี Property 'CLOUDY_PIXEL_PERCENTAGE')

source code:

```
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter6.md#4-filter
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

// Filter by cloudy pixel percentage
var collection = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-01-01', '2025-03-31') // Filter method by date
  .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20)) // Filter method by
property
  .filterBounds(polygon) // Filter method by bounds
Map.centerObject(polygon, 10);
Map.addLayer(collection, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'Image
Collection');
```

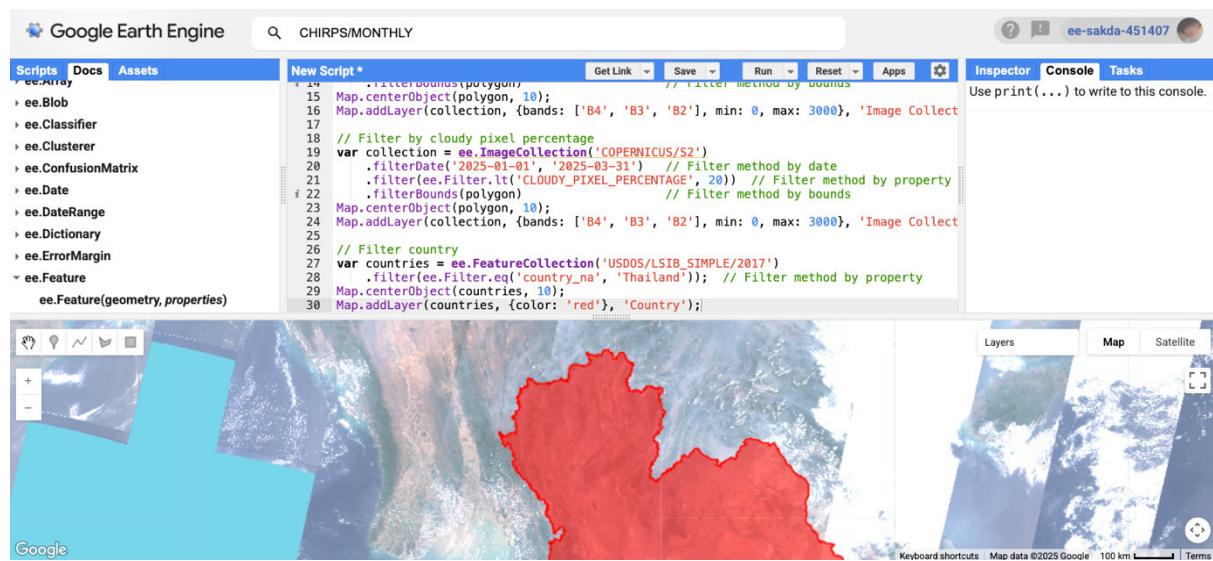
การใช้ ee.Filter ใน GEE มีกลุ่มของพังก์ชัน ee.Filter ที่ยืดหยุ่นกว่าในการสร้างเงื่อนไขการกรองที่ซับซ้อน เช่น

- ee.Filter.lt(propertyName, value): น้อยกว่า (less than)
- ee.Filter.lte(propertyName, value): น้อยกว่าหรือเท่ากับ (less than or equal to)
- ee.Filter.gt(propertyName, value): มากกว่า (greater than)
- ee.Filter.gte(propertyName, value): มากกว่าหรือเท่ากับ (greater than or equal to)
- ee.Filter.eq(propertyName, value): เท่ากับ (equal to)
- ee.Filter.neq(propertyName, value): ไม่เท่ากับ (not equal to)
- ee.Filter.inList(propertyName, list): Property อยู่ใน List ที่กำหนด
- ee.Filter.stringContains(propertyName, string): Property ที่เป็น String มีข้อความที่กำหนด
- การนำ Filter ไปใช้กับ Collection ใช้เมธอด .filter() ของ ImageCollection

ตัวอย่างด้านล่างเป็นการเลือกข้อมูลขอบเขตประเทศไทยโดยเมธอด filter

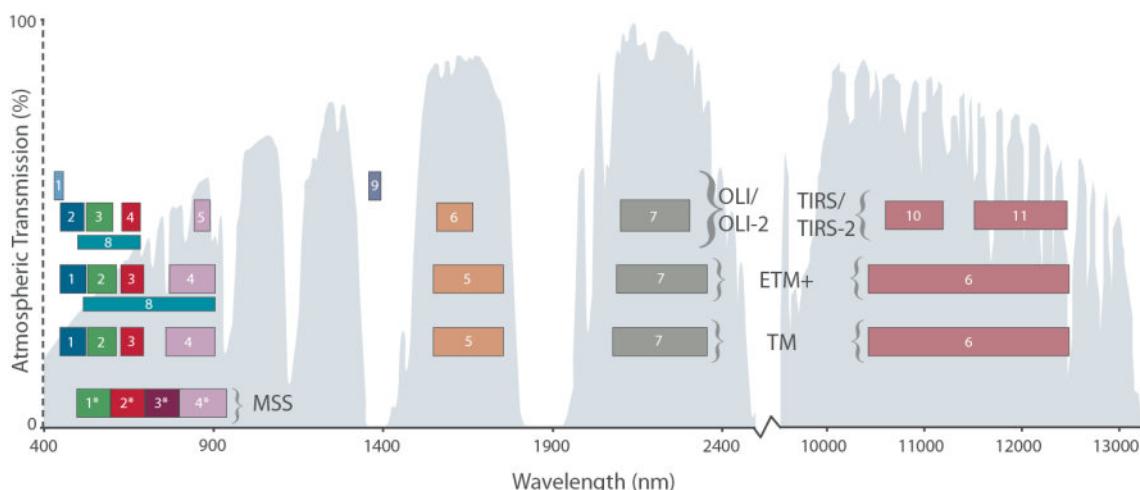
```
// Filter country
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017')
  .filter(ee.Filter.eq('country_na', 'Thailand')) // Filter method by property
Map.centerObject(countries, 10);
Map.addLayer(countries, {color: 'red'}, 'Country');
```

ผลลัพธ์การทำงานของโคด



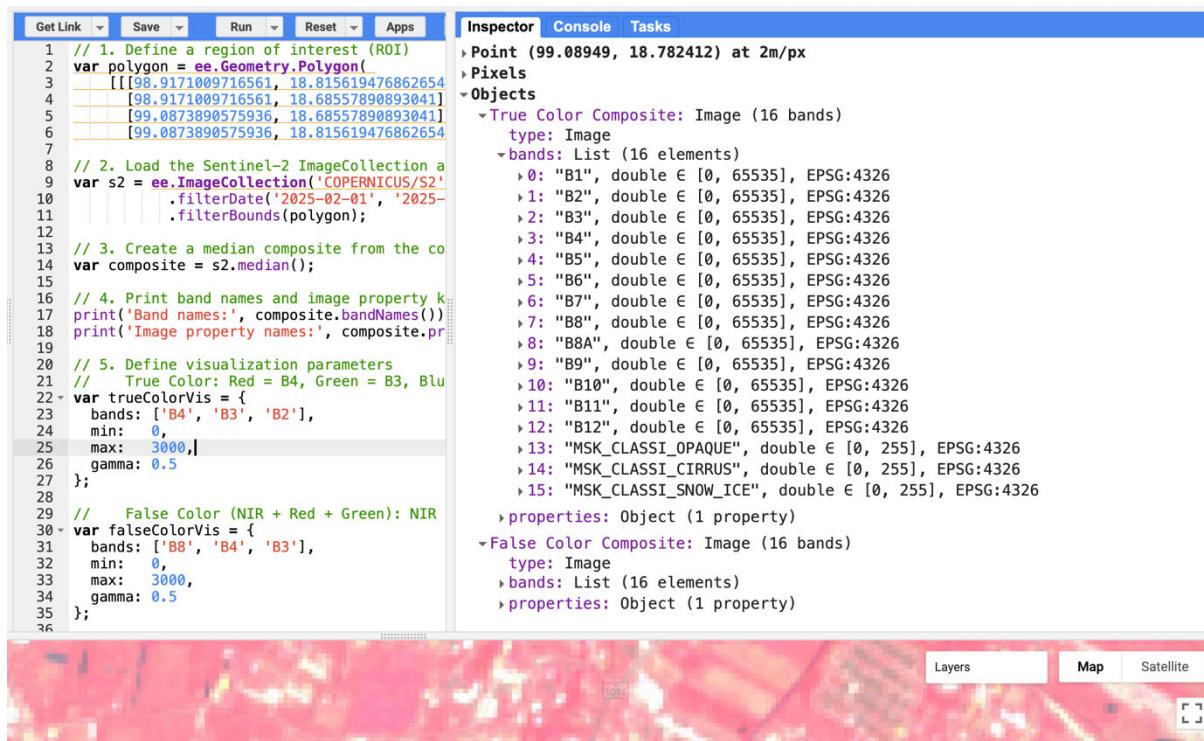
6.5 ช่วงคลื่น (Bands)

Bands คือข้อมูลภาพที่เก็บจากการสะท้อนหรือการแพร่รังสีของพื้นผิวโลกในแต่ละช่วงความยาวคลื่น (เช่น แสงสี น้ำเงิน, เขียว, แดง, อินฟราเรดไกล, อินฟราเรดคลื่นสั้น, อินฟราเรดความร้อน) แต่ละ Band จะมีข้อมูล (Data Type) ของตัวเอง (เช่น int16, float)



ที่มา https://landsat.gsfc.nasa.gov/wp-content/uploads/2020/08/all_Landsat_bands.png

การตรวจสอบ Bands ของ Image: สามารถใช้คำสั่ง print() กับ ee.Image เพื่อดูข้อมูล Bands ที่มีอยู่ได้ใน Console หรือใช้ Inspector tool เพื่อดูค่าของแต่ละ Band ณ จุดที่คลิกบนแผนที่



```

Get Link Save Run Reset Apps
Inspector Console Tasks
Point (99.08949, 18.782412) at 2m/px
Pixels
Objects
  - True Color Composite: Image (16 bands)
    type: Image
    - bands: List (16 elements)
      0: "B1", double ∈ [0, 65535], EPSG:4326
      1: "B2", double ∈ [0, 65535], EPSG:4326
      2: "B3", double ∈ [0, 65535], EPSG:4326
      3: "B4", double ∈ [0, 65535], EPSG:4326
      4: "B5", double ∈ [0, 65535], EPSG:4326
      5: "B6", double ∈ [0, 65535], EPSG:4326
      6: "B7", double ∈ [0, 65535], EPSG:4326
      7: "B8", double ∈ [0, 65535], EPSG:4326
      8: "B8A", double ∈ [0, 65535], EPSG:4326
      9: "B9", double ∈ [0, 65535], EPSG:4326
      10: "B10", double ∈ [0, 65535], EPSG:4326
      11: "B11", double ∈ [0, 65535], EPSG:4326
      12: "B12", double ∈ [0, 65535], EPSG:4326
      13: "MSK_CLASSI_OPAQUE", double ∈ [0, 255], EPSG:4326
      14: "MSK_CLASSI_CIRRUS", double ∈ [0, 255], EPSG:4326
      15: "MSK_CLASSI_SNOW_ICE", double ∈ [0, 255], EPSG:4326
    - properties: Object (1 property)
  - False Color Composite: Image (16 bands)
    type: Image
    - bands: List (16 elements)
    - properties: Object (1 property)

Layers Map Satellite

```

การเลือก Bands ใช้ฟังก์ชัน `image.select()` สำหรับเลือก Bands ที่ต้องการจาก `ee.Image` เพื่อนำไปใช้ในการวิเคราะห์หรือแสดงผลต่อไป การเลือกเฉพาะ Bands ที่จำเป็นจะช่วยลดปริมาณข้อมูลที่ต้องประมวลผล ตัวอย่างนี้เป็นการ Bands มาใช้แสดงผลแบบ ผสมสีจริง (True color composite) สีผสมเท็จ (False color composite)

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#create-a-median-composite-and-select-bands

```

//Define a region of interest (ROI)
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
  [98.9171009716561, 18.68557890893041],
  [99.0873890575936, 18.68557890893041],
  [99.0873890575936, 18.815619476862654]]]);

// Load the Sentinel-2 ImageCollection and filter by date & ROI
var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon);

// Create a median composite from the collection
var composite = s2.median();

// Print band names and image property keys to the Console
print('Band names:', composite.bandNames());
print('Image property names:', composite.propertyNames());

```

```

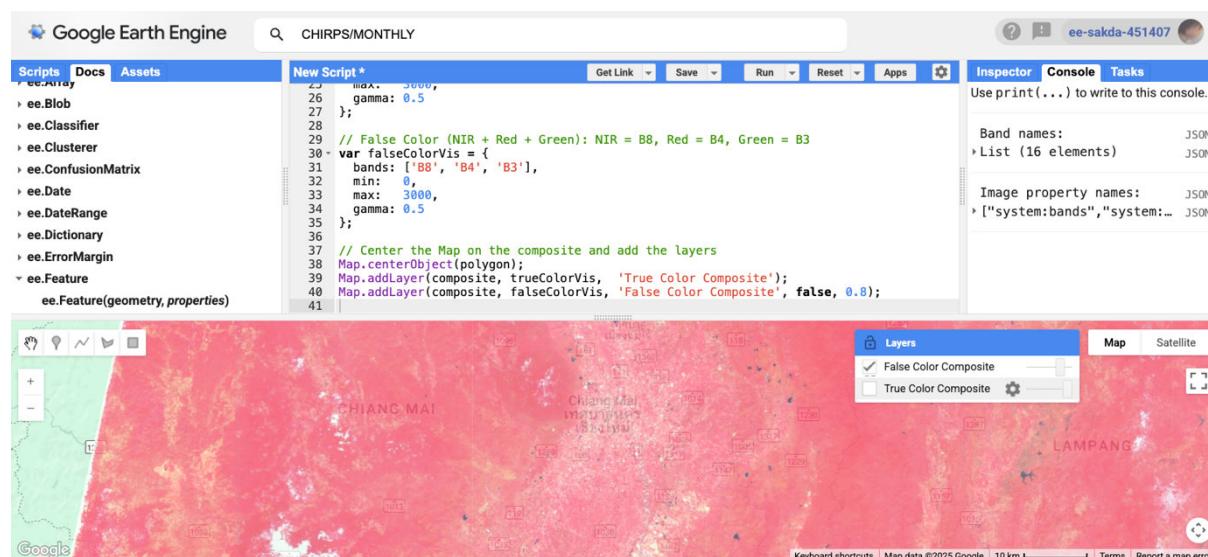
// Define visualization parameters
// True Color: Red = B4, Green = B3, Blue = B2
var trueColorVis = {
  bands: ['B4', 'B3', 'B2'],
  min: 0,
  max: 3000,
  gamma: 0.5
};

// False Color (NIR + Red + Green): NIR = B8, Red = B4, Green = B3
var falseColorVis = {
  bands: ['B8', 'B4', 'B3'],
  min: 0,
  max: 3000,
  gamma: 0.5
};

// Center the Map on the composite and add the layers
Map.centerObject(polygon);
Map.addLayer(composite, trueColorVis, 'True Color Composite');
Map.addLayer(composite, falseColorVis, 'False Color Composite', false, 0.8);

```

ผลลัพธ์การทำงานของโคด



การเลือก Band สามารถทำได้หลายวิธี ดังนี้

- การเลือก Band เดียว เช่น var blueBand = image.select('B2');
- การเลือกหลาย Bands (ใช้ List) เช่น var rgbBands = image.select(['B4', 'B3', 'B2']);
- การเปลี่ยนชื่อ Bands สามารถเปลี่ยนชื่อ Bands ได้พร้อมกับการเลือก

source code:

```
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter6.md#band-selection-and-add-to-new-object
// band selection and add to new object
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
// Load the Sentinel-2 ImageCollection and filter by date & ROI
var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon);
// Create a median composite from the collection
var composite = s2.median();
// Select single bands
var band4 = composite.select('B4');
var band3 = composite.select('B3');
var band2 = composite.select('B2');
// rename bands
band4 = band4.rename('Red');
band3 = band3.rename('Green');
band2 = band2.rename('Blue');
var rgbSingleBand = band4.addBands(band3).addBands(band2);
// Select multiple bands name
var rgbMultiBand = composite.select(['B4', 'B3', 'B2']);
Map.centerObject(polygon, 10);
Map.addLayer(rgbSingleBand, {min: 0, max: 3000}, 'RGB select from single band');
Map.addLayer(rgbMultiBand, {min: 0, max: 3000}, 'RGB select from multiple band');
```

6.6 การคำนวณทางคณิตศาสตร์

GEE มีฟังก์ชันทางคณิตศาสตร์มากมายที่สามารถนำมาใช้กับ ee.Image ได้โดยตรง การคำนวณเหล่านี้จะถูกนำไปใช้กับทุก Pixel ในภาพ (Pixel-wise operation) หรือระหว่าง Bands ของภาพ

6.6.1 การคำนวณระหว่าง Bands

ตัวอย่างนี้ใช้ดัชนีพืชพรรณ (Vegetation Indices) เป็นการคำนวณที่ใช้ค่าการสะท้อนแสงจาก Bands ต่างๆ เพื่อ弄ซึ่งปริมาณหรือความเข้มของพืชพรรณ ดัชนีที่นิยมใช้กันมากที่สุดคือ NDVI (Normalized Difference Vegetation Index) โดยมีสูตร ดังนี้

$$\text{NDVI: NDVI} = (\text{NIR} - \text{Red}) / (\text{NIR} + \text{Red})$$

* NIR คือค่าการสะท้อนแสงในช่วงคลื่นอินฟราเรดใกล้

* Red คือค่าการสะท้อนแสงในช่วงคลื่นแสงสีแดง

วิธีที่ 1 ใช้ฟังก์ชันทางคณิตศาสตร์พื้นฐาน

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#calculate-ndvi-using-band-math

```
// Calculate NDVI using band math
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
  
var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon);

// Create a median composite from the collection
var composite = s2.median();
// Select bands for NDVI calculation
var nirBand = composite.select('B8'); // NIR band
var redBand = composite.select('B4'); // Red band

Map.centerObject(polygon, 10);
Map.addLayer(composite, {bands: ['B4', 'B3', 'B2'], min: 0, max: 3000}, 'RGB Composite');

// Calculate NDVI
var ndvi = nirBand.subtract(redBand).divide(nirBand.add(redBand)).rename('NDVI');

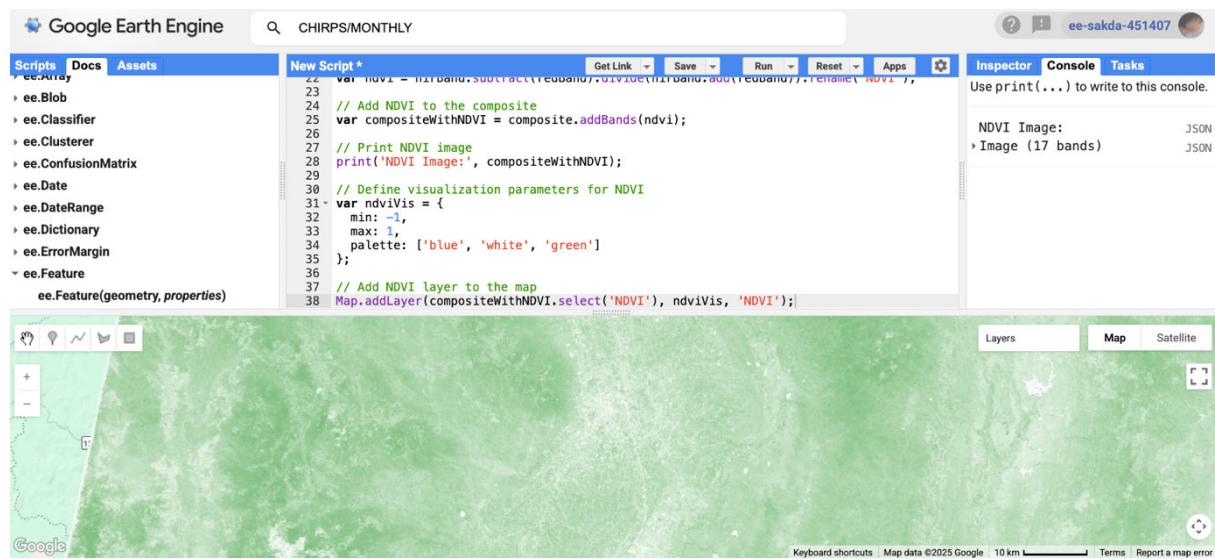
// Add NDVI to the composite
var compositeWithNDVI = composite.addBands(ndvi);

// Print NDVI image
print('NDVI Image:', compositeWithNDVI);

// Define visualization parameters for NDVI
var ndviVis = {
  min: -1,
  max: 1,
  palette: ['blue', 'white', 'green']
};

// Add NDVI layer to the map
Map.addLayer(compositeWithNDVI.select('NDVI'), ndviVis, 'NDVI');
```

ผลลัพธ์การทำงานของโค้ด



The screenshot shows the Google Earth Engine (GEE) web interface. At the top, there's a navigation bar with 'Google Earth Engine' and a search bar containing 'CHIRPS/MONTHLY'. Below the search bar is a script editor titled 'New Script *' with the following code:

```

22 var ndvi = nirband.subtract(redband).divide(nirband.add(redband)).rename('NDVI');
23 // Add NDVI to the composite
24 var compositeWithNDVI = composite.addBands(ndvi);
25
26 // Print NDVI image
27 print('NDVI Image:', compositeWithNDVI);
28
29 // Define visualization parameters for NDVI
30 var ndviVis = {
31   min: -1,
32   max: 1,
33   palette: ['blue', 'white', 'green']
34 };
35
36
37 // Add NDVI layer to the map
38 Map.addLayer(compositeWithNDVI.select('NDVI'), ndviVis, 'NDVI');

```

To the right of the script editor is the 'Inspector' panel showing 'NDVI Image: Image (17 bands)' and the 'Console' panel which displays 'Use print(...) to write to this console.' Below the script editor is a map view showing a green landscape. The bottom right corner of the map view includes standard map controls like 'Layers', 'Map', and 'Satellite'.

วิธีที่ 2 ใช้ฟังก์ชัน normalizedDifference()

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#calculate-ndvi-using-normalizeddifference-method

```

// Calculate NDVI using normalizedDifference() method
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon);
// Create a median composite from the collection
var composite = s2.median();
// Select bands for NDVI calculation
var nirBand = composite.select('B8'); // NIR band
var redBand = composite.select('B4'); // Red band
// normalizedDifference() method
var ndvi = composite.normalizedDifference(['B8', 'B4']).rename('NDVI');
// Add NDVI to the composite
var compositeWithNDVI = composite.addBands(ndvi);
// Print NDVI image
print('NDVI Image:', compositeWithNDVI);
// Define visualization parameters for NDVI
var ndviVis = {
  min: -1,
  max: 1,
  palette: ['blue', 'white', 'green']
};
// Add NDVI layer to the map
Map.addLayer(compositeWithNDVI.select('NDVI'), ndviVis, 'NDVI');

```

นอกจาก NDVI ยังมีดัชนีอื่นๆ อีกมากmanyที่สามารถคำนวณได้ เช่น

- EVI (Enhanced Vegetation Index) ปรับปรุงจาก NDVI เพื่อลดผลกระทบจากบริการอากาศและพืช
หลังของดิน
- NDWI (Normalized Difference Water Index) ใช้ตรวจสอบแหล่งน้ำ
- NDBI (Normalized Difference Built-up Index) ใช้ตรวจสอบพื้นที่สิ่งปลูกสร้าง

สามารถศึกษาเพิ่มเติม ดังนี้ <https://www.indexdatabase.de/>

6.6.2 การใช้ฟังก์ชันทางคณิตศาสตร์

1) ฟังก์ชันพื้นฐานของ ee.Image

รองรับการดำเนินการทางคณิตศาสตร์พื้นฐานโดยตรง ดังตัวอย่างการวิเคราะห์ NDVI ที่ผ่านมา ตัวอย่างของฟังก์ชันพื้นฐาน ee.Image เช่น

- image1.add(image2) หรือ image.add(constant) เป็นการบวก
- image1.subtract(image2) หรือ image.subtract(constant) เป็นการลบ
- image1.multiply(image2) หรือ image.multiply(constant) เป็นการคูณ
- image1.divide(image2) หรือ image.divide(constant) เป็นการหาร
- image.pow(exponent) เป็นการยกกำลัง
- image.sqrt() เป็นรากที่สอง
- image.abs() เป็นค่าสัมบูรณ์

- `image.sin()`, `image.cos()`, `image.tan()` เป็นฟังก์ชันตรีโกณมิติ
- และอื่นๆ อีกมากมาย (ดูเพิ่มเติมใน Docs ใต้ `ee.Image`)

2) ฟังก์ชัน `expression(expressionString, map)`

เป็นฟังก์ชันที่ช่วยให้สามารถเขียนสูตรคำนวณที่ซับซ้อนโดยใช้ String ที่มีตัวแปรแทน Bands และมี map ที่เป็น Dictionary เชื่อมโยงชื่อตัวแปรใน String กับ Bands จริงใน ee.Image ตัวอย่างด้านล่าง เป็นการใช้ `expression()` ทำให้อ่านและทำความเข้าใจสูตรคำนวณได้ง่าย กว่าการเขียนแบบ band math โดยเฉพาะสูตรที่มีหลาย Bands และค่าคงที่เข้ามาเกี่ยวข้อง

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#calculate-evi-using-imageexpression-and-band-math

```
// EVI calculation using image.expression() and band math
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon);
// Create a median composite from the collection
var composite = s2.median();

// Select bands for EVI calculation
var nirBand = composite.select('B8'); // NIR band
var redBand = composite.select('B4'); // Red band
var blueBand = composite.select('B2'); // Blue band

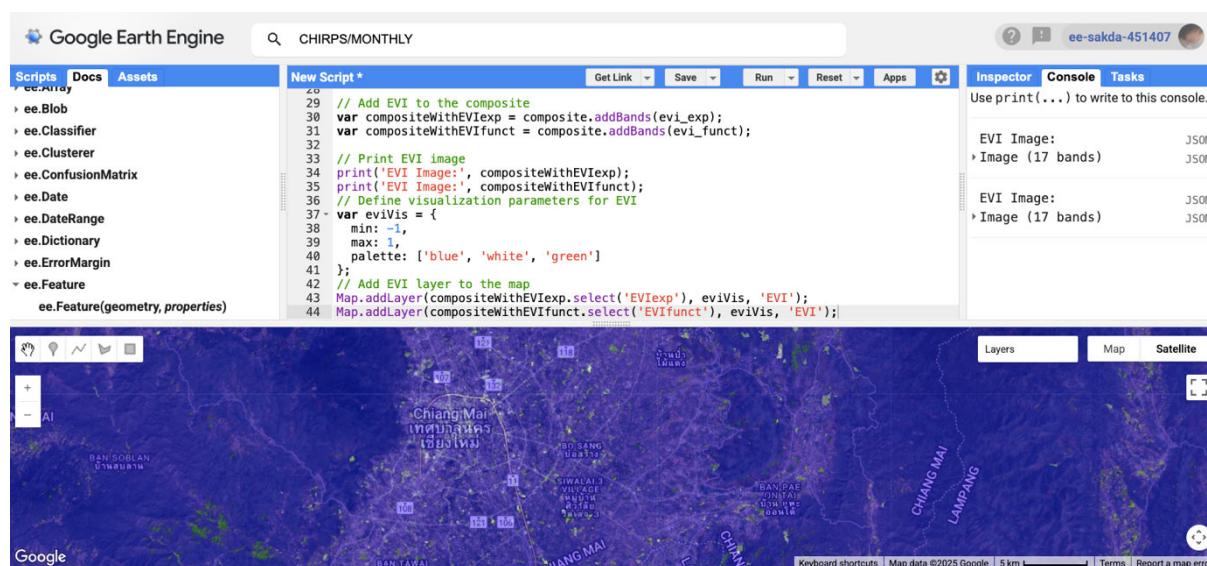
// Calculate EVI with image.expression
var evi_exp = composite.expression(
  '2.5 * ((NIR - RED) / (NIR + 6 * RED - 7.5 * BLUE + 1))', {
    'NIR': nirBand,
    'RED': redBand,
    'BLUE': blueBand
  }).rename('EVIexp');

// Calculate EVI with band math
var evi_func = nirBand.subtract(redBand).divide(nirBand.add(redBand.multiply(6)).subtract(blueBand.multiply(7.5)).add(1)).multiply(2.5).rename('EVIfunc');

// Add EVI to the composite
var compositeWithEVIexp = composite.addBands(evi_exp);
var compositeWithEVIfunc = composite.addBands(evi_func);
```

```
// Print EVI image
print('EVI Image:', compositeWithEVIexp);
print('EVI Image:', compositeWithEVIfunct);
// Define visualization parameters for EVI
var eviVis = {
  min: -1,
  max: 1,
  palette: ['blue', 'white', 'green']
};
// Add EVI layer to the map
Map.addLayer(compositeWithEVIexp.select('EVIexp'), eviVis, 'EVI');
Map.addLayer(compositeWithEVIfunct.select('EVIfunct'), eviVis, 'EVI');
```

ผลลัพธ์การทำงานของโค้ด



6.7 การตัดภาพตามขอบเขตพื้นที่สนใจ (Clipping)

การวิเคราะห์ข้อมูลเฉพาะภายในขอบเขตพื้นที่ที่กำหนด เช่น ขอบเขตจังหวัด พื้นที่ลุ่มน้ำ หรือพื้นที่ศึกษาที่คาดขึ้นเอง GEE อนุญาตให้เราตัด (Clip) ee.Image ให้เหลือเฉพาะส่วนที่อยู่ในขอบเขตนั้นๆ ซึ่งเราอาจใช้เครื่องมือวัด Point, Line, หรือ Polygon ใน Code Editor เพื่อสร้างขอบเขตพื้นที่ศึกษา หรือจากอัปโหลด shapfile เข้าไปยัง asset

ตัวอย่างด้านล่างใช้ฟังก์ชัน image.clip(geometry) ใช้สำหรับตัด ee.Image ตามขอบเขตของ ee.Geometry ที่กำหนด Pixel ที่อยู่นอก Geometry จะถูก Mask ออกไป (ถ้ายังเป็น NoData) Geometry ที่จะใช้เป็นขอบเขตในการตัด สามารถเป็นไปได้ทั้ง ee.Geometry, ee.Feature, หรือ ee.FeatureCollection (ถ้าเป็น Feature หรือ FeatureCollection, GEE จะใช้ Union ของ Geometries ทั้งหมด)

```

source code:
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter6.md#clipping-an-image-with-a-polygon

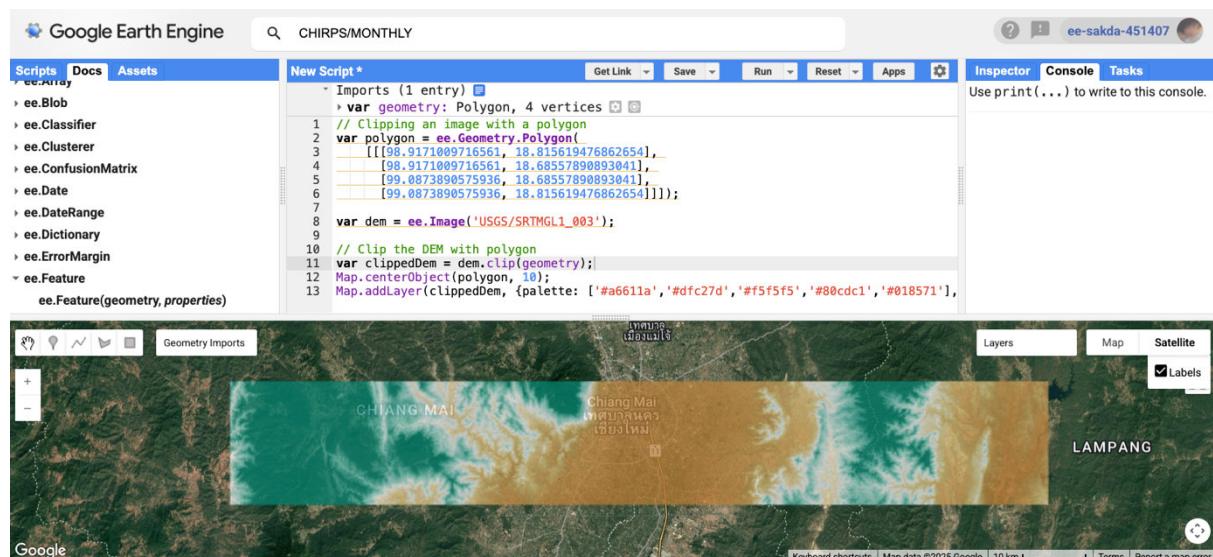
// Clipping an image with a polygon
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

var dem = ee.Image('USGS/SRTMGL1_003');

// Clip the DEM with polygon
var clippedDem = dem.clip(polygon);
Map.centerObject(polygon, 10);
Map.addLayer(clippedDem, {palette:
  ['#a6611a','#dfc27d','#f5f5f5','#80cdc1','#018571'], min: 250, max: 1000}, 'Clipped DEM');

```

ผลลัพธ์การทำงานของโค้ด



สำหรับการตัด ImageCollection หากต้องการตัดทุกภาพใน ImageCollection สามารถใช้ map() ร่วมกับ clip() ดังนี้

```

source code:
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter6.md#clipping-an-image-collection-with-a-polygon

// Clipping an image collection with a polygon
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);

```

```

function clipImage(image) {
  return image.clip(polygon);
}
var clippedComposite = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon)
  .map(clipImage)
  .median()
Map.centerObject(polygon, 10);
Map.addLayer(clippedComposite, {bands: ['B4', 'B3', 'B2'], min: 500, max: 3000},
'Clipped Composite');

```

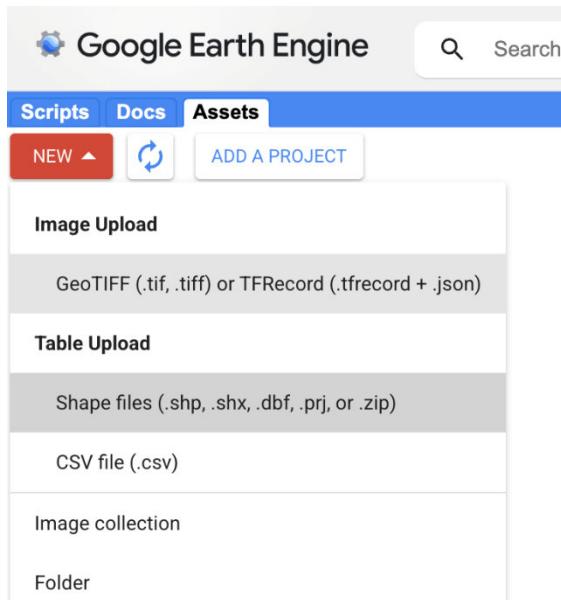
ผลลัพธ์การทำงานของโคด

The screenshot shows the Google Earth Engine interface. On the left, the 'Assets' tab is selected, displaying a list of Earth Engine objects like ee.Array, ee.Blob, ee.Classifier, etc. In the center, the 'New Script' pane contains the provided JavaScript code. On the right, the 'Map' viewer shows a satellite image of a forested area in LAMPANG, Thailand, with a specific polygon highlighted in blue. The map includes various road numbers and labels like 'เมืองแม่จัน' and 'LAMPANG'. The bottom status bar indicates 'Keyboard shortcuts' and 'Map data ©2023 Google'.

6.8 การนำเข้าข้อมูล (Importing Data)

การนำเข้าข้อมูล Vector สามารถทำได้โดยการอัปโหลดข้อมูลไปยัง Assets

- 1) ไปที่แท็บ "Assets" ทางด้านซ้ายของ Code Editor
- 2) คลิกปุ่ม "NEW" > "Table upload"



- 3) เลือกประเภท Source file
- 4) กำหนดชื่อขั้นตอน > Upload

Upload a new shapefile asset

Source files

SELECT

Please drag and drop or select files for this asset.
Allowed extensions: shp, zip, dbf, prj, shx, cpg, fix, qix, sbn or shp.xml.

cm_province_4326.cpg	trash
cm_province_4326.dbf	trash
cm_province_4326.prj	trash
cm_province_4326.shp	trash
cm_province_4326.shx	trash

Asset ID

Asset Name: projects/ee-sakda-451407/assets/ cm_province_4326

Properties

Metadata properties about the asset which can be edited during asset upload and after ingestion. The "system:time_start" property is used as the primary date of the asset.

Add start time | Add end time | Add property

Advanced options

Character encoding: UTF-8

CANCEL UPLOAD

5) ที่ Tab Tasks จะแสดงความคืบหน้าของการ upload ข้อมูล

The screenshot shows the 'Tasks' tab selected in the top navigation bar. A single task is listed: 'Ingest table: "projects/ee-sakda-451407/assets/cm_province_4326"' with a checkmark and a timestamp of '<1m'. Below this, there's a search bar and a section for 'SUBMITTED TASKS'.

6) เมื่ออัปโหลดสำเร็จข้อมูลจะถูกแสดงใน Assets

The screenshot shows the 'Assets' tab selected in the top navigation bar. It displays two sections: 'CLOUD ASSETS' containing a folder 'ee-sakda-451407' with files 'fire' and 'cm_province_4326'; and 'LEGACY ASSETS' containing a file 'users/sakdahomhuan'.

ประเภทของข้อมูล Vector ที่สามารถนำเข้า asset ได้ ประกอบด้วย

- Shape files (.shp, .dbf, .shx, และไฟล์อื่นๆ ที่เกี่ยวข้อง): ต้องอัปโหลดไฟล์ทั้งหมดที่ประกอบกันเป็น Shapefile (อย่างน้อย .shp, .shx, .dbf และ .prj ถ้ามี) โดยสามารถ Zip ไฟล์ทั้งหมดรวมกันแล้วอัปโหลดไฟล์ Zip นั้นได้
- CSV (.csv): ไฟล์ CSV จะต้องมีคอลัมน์สำหรับ Latitude และ Longitude (หรือ WKT - Well-Known Text สำหรับ Geometry) เพื่อให้ GEE สร้าง Geometry ได้ หรือสามารถอัปโหลดเป็น Table ที่ไม่มี Geometry ก็ได้

เมื่อข้อมูลถูกอัปโหลดไปยัง Assets เรียบร้อยแล้ว สามารถเรียกใช้ในสคริปต์โดยตรงโดยใช้ Asset ID

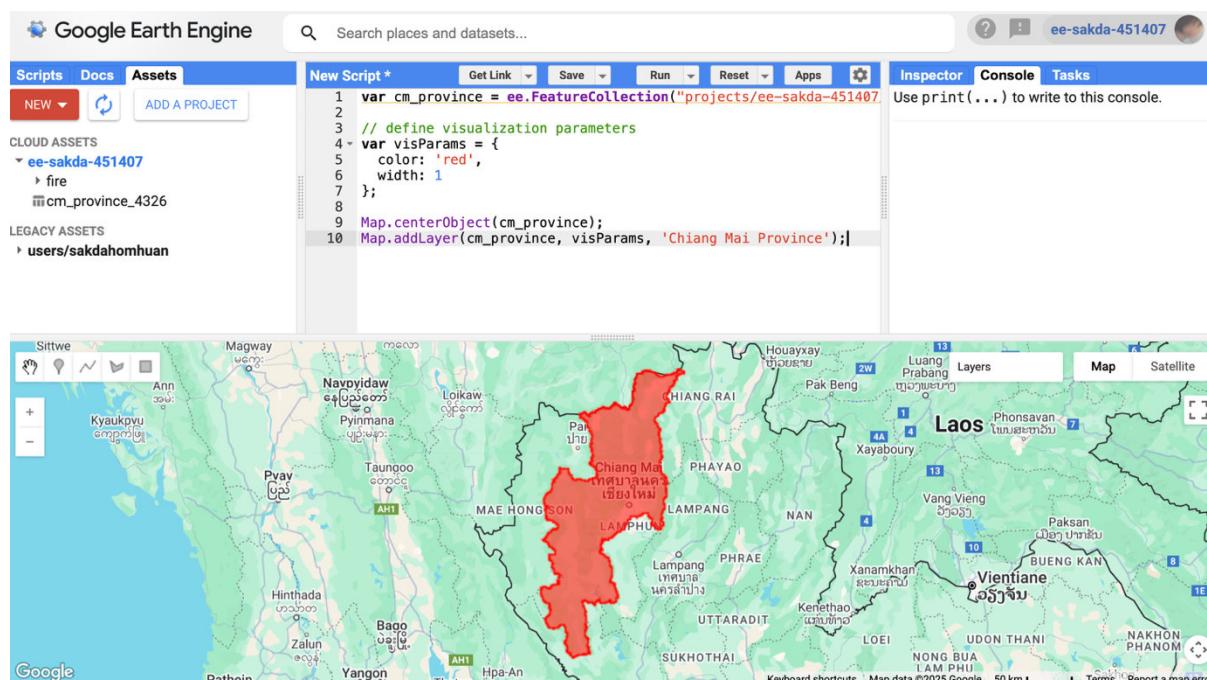
source code:

```
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter6.md#clipping-an-image-collection-with-a-polygon
var cm_province = ee.FeatureCollection("projects/ee-sakda-451407/assets/cm_province_4326");

// define visualization parameters
var visParams = {
  color: 'red',
  width: 1
};

Map.centerObject(cm_province);
Map.addLayer(cm_province, visParams, 'Chiang Mai Province');
```

ผลลัพธ์การทำงานของโค้ด



6.9 การส่งออกข้อมูล (Exporting Data)

การส่งออกข้อมูลเป็นขั้นตอนสำคัญหลังจากการวิเคราะห์หรือประมวลผลข้อมูลภาพและข้อมูลอื่นๆ เสร็จสิ้น เพื่อนำผลลัพธ์ไปใช้งานต่อในซอฟต์แวร์อื่น ในตัวอย่างนี้เป็นการส่งออกข้อมูลภาพจากดาวเทียมหรือผลลัพธ์จากการประมวลผลภาพ เช่น ภาพดัชนีพืชพรรณ (NDVI), ภาพผลลัพธ์การจำแนกประเภท (Classification Map) หรือภาพผลต่าง (Difference Image) จากการตรวจจับการเปลี่ยนแปลง ดังนี้

6.9.1 การส่งออกข้อมูล Images ไปยัง Google Drive

เป็นวิธีที่สะดวกในการบันทึกไฟล์ภาพผลลัพธ์ขนาดใหญ่ลงในบัญชี Google Drive สำหรับหรือขององค์กร เพื่อนำไปเปิดใช้งานต่อในซอฟต์แวร์ GIS บนเครื่องคอมพิวเตอร์ (เช่น QGIS, ArcGIS) ทำการวิเคราะห์เพิ่มเติม หรือสำรองข้อมูล โดยใช้ฟังก์ชันการส่งออก เช่น Export.image.toDrive()

```
Export.image.toDrive(image, description, folder, fileNamePrefix, dimensions, region, scale, crs,
crsTransform, maxPixels, shardSize, fileDimensions, skipEmptyTiles, fileFormat, formatOptions, priority)
```

โดยต้องกำหนดพารามิเตอร์ที่สำคัญดังตาราง 4

ตารางที่ 4 พารามิเตอร์การส่งออกข้อมูล Image เข้าสู่ Google Drive

พารามิเตอร์	ประเภทข้อมูล	คำอธิบาย
image	ee.Image	ภาพแรสเตอร์ที่ต้องการส่งออก
description	String	ชื่อคำอธิบาย (task name) ที่ปรากฏใน Tasks panel ของ Code Editor
folder (optional)	String	ชื่อโฟลเดอร์บน Google Drive ที่จะเก็บไฟล์ผลลัพธ์ (ถ้าไม่ระบุ จะเก็บใน root)
fileNamePrefix	String	คำนำหน้าชื่อไฟล์ผลลัพธ์ (เช่น "my_export" จะได้ไฟล์ my_export.tif)
region	Geometry List<Number>	ขอบเขตพื้นที่ส่งออกเป็น ee.Geometry หรือ array ในรูปแบบ [xmin, ymin, xmax, ymax] (ค่าพิกัด lon/lat ตาม CRS ที่กำหนด)
scale (optional)	Number	ความละเอียดเชิงพื้นที่ (เมตรต่อพิกเซล)
crs (optional)	String	ระบบพิกัดอ้างอิง (Coordinate Reference System) เช่น "EPSG:4326"
crsTransform (optional)	List<Number>	ระบบพิกัดอ้างอิงที่ต้องการเปลี่ยน
maxPixels (optional)	Number	จำนวนพิกเซลสูงสุดที่อนุญาตให้ส่งออก (ถ้าเกินจะ error) ค่าเริ่มต้น คือ 1e13
fileFormat (optional)	String	รูปแบบไฟล์ที่ต้องการส่งออก เช่น "GeoTIFF", "TFRecord" (default "GeoTIFF")
formatOptions (optional)	Object	ออบชันเฉพาะของรูปแบบไฟล์ เช่น {cloudOptimized: true} สำหรับ GeoTIFF แบบ CO-TIFF
skipEmptyTiles (optional)	Boolean	กำหนดให้ข้าม tile ที่ไม่มีข้อมูล (true) หรือไม่ (false) ช่วยลดขนาดไฟล์ที่ export เป็น tiles
shardSize (optional)	Number	ขนาด tile ในหน่วยพิกเซล (ใช้เมื่อ export เป็น tiled output)
pyramidingPolicy (optional)	Object	การลดทอน (reducer) สำหรับแต่ละแบบ เมื่อสร้าง overviews/pyramids เช่น {B4: 'mode', B3: 'mean'}

ตัวอย่างการใช้งาน

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#91-export-an-image-to-google-drive

```
// Export an image to Google Drive
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon);
var composite = s2.median();
// Select bands for NDVI calculation
var nirBand = composite.select('B8'); // NIR band
var redBand = composite.select('B4'); // Red band
// Calculate NDVI
var ndvi = composite.normalizedDifference(['B8', 'B4']).rename('NDVI');
// Add NDVI to the composite
var compositeWithNDVI = composite.addBands(ndvi);
// Export the NDVI image to Google Drive
Export.image.toDrive({
  image: compositeWithNDVI.select('NDVI'),
  description: 'NDVI_Export',
  scale: 30,
  region: polygon,
  maxPixels: 1e13
});
```

ผลลัพธ์การทำงานของโคด

The screenshot shows the Google Earth Engine interface. On the left, the 'New Script' pane displays the provided JavaScript code. On the right, the 'Inspector' and 'Tasks' panes are visible. The 'Tasks' pane shows a single task named 'NDVI_Export' with a status of 'RUNNING'. A message above the tasks says 'UPDATING...'. In the bottom right corner of the interface, there is a timestamp '1m'.

```
1 // Export an image to Google Drive
2 var polygon = ee.Geometry.Polygon(
3   [[[98.9171009716561, 18.815619476862654],
4     [98.9171009716561, 18.68557890893041],
5     [99.0873890575936, 18.68557890893041],
6     [99.0873890575936, 18.815619476862654]]]);
7 var s2 = ee.ImageCollection('COPERNICUS/S2')
8   .filterDate('2025-02-01', '2025-06-28')
9   .filterBounds(polygon);
10 var composite = s2.median();
11 // Select bands for NDVI calculation
12 var nirBand = composite.select('B8'); // NIR band
13 var redBand = composite.select('B4'); // Red band
14 // Calculate NDVI
15 var ndvi = composite.normalizedDifference(['B8', 'B4']).rename('NDVI');
16 // Add NDVI to the composite
17 var compositeWithNDVI = composite.addBands(ndvi);
18 // Export the NDVI image to Google Drive
19 Export.image.toDrive({
20   image: compositeWithNDVI.select('NDVI'),
21   description: 'NDVI_Export',
22   scale: 30,
23   region: polygon,
24   maxPixels: 1e13
25});
```

ที่ Tab Task

- 1) คลิกปุ่ม RUN
- 2) กำหนดรายละเอียดสำหรับจัดเก็บข้อมูล

Task: Initiate image export

Task name (no spaces) *

NDVI_Export

Coordinate Reference System (CRS)

EPSG:3857

Scale (m/px)

30

DRIVE CLOUD STORAGE EE ASSET

Drive folder
gistnorth_gee

Filename *

NDVI_Export

File format *

GEO_TIFF

CANCEL **RUN**

6.9.2 การส่งออกข้อมูล Table ไปยัง Google Drive

ในการส่งออกข้อมูลเชิงพื้นที่ (FeatureCollection) จาก Google Earth Engine ไปยัง Google Drive ในรูปแบบไฟล์ Shapefile (SHP) ให้ทำตามขั้นตอนดังนี้

`Export.table.toDrive(collection, description, folder, fileNamePrefix, fileFormat, selectors, maxVertices, priority)`

พารามิเตอร์การส่งออกข้อมูล FeatureCollection เข้าสู่ Google Drive

พารามิเตอร์	ประเภทข้อมูล	คำอธิบาย
collection	ee.FeatureCollection	ชุดข้อมูลฟีเจอร์ (FeatureCollection) ที่ต้องการส่งออก
description	String (optional)	ชื่อคำอธิบาย (task name) ที่แสดงใน Tasks panel; ค่าเริมต้นคือ "myExportTableTask"
folder	String (optional)	ชื่อโฟลเดอร์บน Google Drive สำหรับเก็บไฟล์ผลลัพธ์; ถ้าไม่ระบุ จะเก็บไว้ใน Drive root

fileNamePrefix	String (optional)	คำนำหน้าชื่อไฟล์ผลลัพธ์ (เช่น "my_export" จะได้ไฟล์ my_export.*); ค่าเริ่มต้น คือค่าของ description
fileFormat	String (optional)	รูปแบบไฟล์ที่ต้องการส่งออก: "CSV" (default), "GeoJSON", "KML", "KMZ", "SHP", หรือ "TFRecord"
selectors	List<String> (optional)	รายชื่อคุณสมบัติ (properties) ที่ต้องการส่งออก; หากไม่มีระบุ จะส่งออกทุกพิล็อตของแต่ละ Feature
maxVertices	Number (optional)	จำนวนจุดมุม (vertices) สูงสุดต่อ geometry หนึ่งชิ้น; หาก geometry มีหลายกว่า จะถูกตัดแบ่งให้เหลือน้อยกว่า ค่านี้ ก่อนส่งออก
priority	Number (optional)	ลำดับความสำคัญของ task (0-9999); ค่าสูง task จะถูกจัดคิวไว้ยิ่งขึ้น; ค่าเริ่มต้นคือ 100

ตัวอย่างการใช้งาน

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#92-export-a-feature-collection-to-google-drive

```
// Get the border of Thailand
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');
var thailandBorder = countries.filter(ee.Filter.eq('country_na', 'Thailand'));
// Export the feature collection to Google Drive
Export.table.toDrive({
  collection: thailandBorder,
  description: 'CM_Province_Export',
  fileFormat: 'SHP'
});
```

ที่ Tab Task

- 1) คลิกปุ่ม RUN
- 2) กำหนดรายละเอียดสำหรับจัดเก็บข้อมูล

Task: Initiate table export

Task name (no spaces) *

DRIVE CLOUD STORAGE EE ASSET FEATURE VIEW ASSET BIGQUERY

Drive folder

Filename *

File format *

CANCEL RUN

6.9.3 การส่งออกไปยัง GEE Assets

GEE Assets คือพื้นที่จัดเก็บข้อมูลส่วนตัวหรือที่แชร์กันภายในแพลตฟอร์ม Google Earth Engine การส่งออกไปยัง Assets มีประโยชน์เมื่อต้องการนำผลลัพธ์ภาพนั้นกลับมาใช้ในการวิเคราะห์หรือประมวลผลอื่นๆ ภายใน GEE ในอนาคตได้อย่างรวดเร็ว

ตารางที่ 5 พารามิเตอร์การส่งออกข้อมูล image เข้าสู่ assets

พารามิเตอร์	ประเภทข้อมูล	คำอธิบาย
image	ee.Image	ภาพแรสเตอร์ที่ต้องการส่งออกเป็น Asset
description	String	ชื่อ Task ที่จะแสดงใน Tasks panel ของ Code Editor
assetId	String	ไอเดียของ Asset ปลายทางใน Assets ของผู้ใช้ (เช่น "users/yourusername/assetName")
pyramiddingPolicy	Object หรือ String	นโยบายการสร้าง pyramids สำหรับแต่ละแบบ (เช่น {B4:'mean',B3:'mode'}) หรือค่าเดียวสำหรับทุกแบบ ("mean")
region	ee.Geometry หรือ List<Number>	ขอบเขตพื้นที่ที่จะส่งออก (Geometry หรือ [xmin, ymin, xmax, ymax] ใน CRS ตามที่กำหนด)
scale	Number	ความละเอียดเชิงพื้นที่ (ขนาดพิกเซลเป็นเมตรต่อพิกเซล)
crs	String	ระบบพิกัดองวิจ (เช่น "EPSG:4326")
dimensions	String หรือ List<Number>	ขนาดภาพที่ส่งออก เช่น "1024x768" หรือ [1024, 768]
maxPixels	Number	จำนวนพิกเซลสูงสุดท่อนุญาตให้ส่งออก (เกินจะ error; ค่า default 1e13)

ตัวอย่างการใช้งาน

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter6.md#93-export-an-image-to-google-earth-engine-assets

```
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon);
```

```

var composite = s2.median();
// Select bands for NDVI calculation
var nirBand = composite.select('B8'); // NIR band
var redBand = composite.select('B4'); // Red band
// Calculate NDVI
var ndvi = composite.normalizedDifference(['B8', 'B4']).rename('NDVI');
// Add NDVI to the composite
var compositeWithNDVI = composite.addBands(ndvi);
// Export the NDVI image to Google Earth Engine assets
Export.image.toAsset({
  image: compositeWithNDVI.select('NDVI'),
  description: 'NDVI_Export_Asset',
  assetId: 'projects/your_project/assets/NDVI_Export_Asset', // Change
  'your_project' to your project ID
  scale: 30,
  region: polygon,
  maxPixels: 1e13
});

```

ເຊື່ອເລີຍກັບການສ່ວນທີ່ໄປຢັ້ງ Google Drive ຜູ້ໃຊ້ຕອງກຳ "Run" Task ເພື່ອເຮີ່ມກະບວນການ ເມື່ອເສົ້າງສິນ
Image Asset ໄໝຈະປາກູ້ໃນສ່ວນ Assets ຂອງຜູ້ໃຊ້

Task: Initiate image export

The screenshot shows the 'Initiate image export' dialog box. It includes fields for Task name (NDVI_Export_Asset), Coordinate Reference System (EPSG:3857), Scale (30), and a dropdown for Earth Engine Asset (set to 'Other root' with path 'projects/your_project/assets/NDVI_Export'). Pyramiding policy is set to 'MEAN'. At the bottom are 'CANCEL' and 'RUN' buttons.

Task name (no spaces) *

NDVI_Export_Asset

Coordinate Reference System (CRS)

EPSG:3857

Scale (m/px)

30

DRIVE CLOUD STORAGE EE ASSET

Earth Engine Asset...

Other root projects/your_project/assets/NDVI_Export

Pyramiding policy

MEAN

CANCEL RUN

บทที่ 7 การสร้างแอปพลิเคชัน Google Earth Engine (GEE Apps)

หลังจากที่เราได้เรียนรู้การวิเคราะห์ข้อมูลภาพในรูปแบบต่างๆ แล้ว ขั้นตอนต่อไปคือการนำผลการวิเคราะห์เหล่านั้นมาสร้างเป็นแอปพลิเคชันที่ผู้ใช้อื่นๆ สามารถเข้ามารอตอ卜และใช้งานได้โดยไม่จำเป็นต้องมีความรู้ในการเขียนโค้ด Google Earth Engine Apps (GEE Apps) ช่วยให้เราสามารถเผยแพร่องค์ความรู้และเครื่องมือวิเคราะห์ของเราในรูปแบบที่ใช้งานง่ายและเข้าถึงได้ผ่านเว็บเบราว์เซอร์

7.1 GEE Apps คืออะไร

GEE Apps คือเว็บแอปพลิเคชันแบบ *buong tac* (interactive) ที่สร้างขึ้นโดยใช้ Google Earth Engine Code Editor เพื่อแบ่งปันและแสดงผลการวิเคราะห์ข้อมูลภูมิสารสนเทศ ผู้ใช้สามารถโต้ตอบกับแอปพลิเคชันผ่านส่วนติดต่อผู้ใช้ (User Interface - UI) เช่น การเลือกพื้นที่ การปรับช่วงเวลา หรือการเลือกพารามิเตอร์ต่างๆ เพื่อดูผลลัพธ์การวิเคราะห์ที่เปลี่ยนแปลงไปแบบไดนามิก

ประโยชน์ของ GEE Apps คือ เข้าถึงง่าย ผู้ใช้ไม่จำเป็นต้องติดตั้งซอฟต์แวร์ใดๆ หรือเขียนโค้ด GEE เอง สามารถเข้าใช้งานผ่านเว็บเบราว์เซอร์ได้ทันที อีกทั้งเป็นช่องทางที่มีประสิทธิภาพในการเผยแพร่องค์ความรู้ เครื่องมือ หรือผลการวิเคราะห์ให้กับสาธารณะหรือกลุ่มผู้ใช้ที่ต้องการ สามารถแสดงผลข้อมูลแทนที่ กราฟ และสถิติที่เปลี่ยนแปลงตามการป้อนข้อมูลของผู้ใช้ ทำให้การสื่อสารผลลัพธ์น่าสนใจและเข้าใจง่ายขึ้น

เบื้องหลังการประมวลผลเป็นส่วนของศูนย์ GEE ที่ทำงานบนเซิร์ฟเวอร์ของ Google เพื่อประมวลผลข้อมูล ตามคำสั่งที่ได้รับจาก UI และส่งผลลัพธ์กลับมาแสดงผล

7.2 การวางแผนและออกแบบ GEE App

ก่อนเริ่มเขียนโค้ด ควรมีการวางแผนและออกแบบแอปพลิเคชันให้ดี เพื่อให้ตอบโจทย์การใช้งาน โดยอาจมี การกำหนดวัตถุประสงค์และกลุ่มผู้ใช้งาน (Defining Objectives and Target Audience) ดังนี้

- แอปพลิเคชันนี้จะทำอะไร (เช่น แสดงการเปลี่ยนแปลงพื้นที่ป่าไม้ คำนวณต้นที่ความแห้งแล้ง ติดตามคุณภาพน้ำ)
- ใครคือกลุ่มผู้ใช้งานหลัก (เช่น นักวิจัย เจ้าหน้าที่รัฐ เกษตรกร หรือประชาชนทั่วไป)
- ผู้ใช้ต้องการเห็นอะไรหรือทำอะไรได้บ้างกับแอปนี้ (เช่น ดูข้อมูลย้อนหลัง เปรียบเทียบข้อมูลระหว่างพื้นที่ ส่องผลกระทบบางส่วน)
- การตอบคำถามเหล่านี้จะช่วยให้กำหนดขอบเขตและพัฒนาการทำงานของแอปได้ชัดเจน

7.3 ออกแบบส่วนติดต่อผู้ใช้

ขั้นตอนนี้ถือเป็นขั้นตอนที่สำคัญ ควรออกแบบให้ใช้งานง่าย ไม่ซับซ้อนเกินไป จัดกลุ่มองค์ประกอบที่เกี่ยวข้องกันไว้ด้วยกัน และมีลำดับการทำงานที่สมเหตุสมผล องค์ประกอบ UI ที่สำคัญของ GEE ประกอบด้วย

- ui.Map แสดงผลข้อมูลเชิงพื้นที่
- ui.Panel เป็นคอนเทนเนอร์สำหรับจัดกลุ่มวิดเจ็ตอื่นๆ สามารถใช้เป็นແນວความคุมหลักหรือແນວแสดงผลข้อมูล
- ui.Label แสดงข้อความหรือคำอธิบาย
- ui.Button ให้ผู้ใช้คลิกเพื่อสั่งงาน
- ui.Textbox ให้ผู้ใช้ป้อนข้อความหรือตัวเลข
- ui.Select ให้ผู้ใช้เลือกจากการรายการตัวเลือกที่กำหนดไว้
- ui.Slider ให้ผู้ใช้เลือกค่าตัวเลขในช่วงที่กำหนด
- ui.Checkbox ให้ผู้ใช้เลือกเปิด/ปิดตัวเลือก
- ui.Chart แสดงผลกราฟต่างๆ

7.4 ตัวอย่างการพัฒนาส่วนติดต่อผู้ใช้ของ GEE

7.4.1 สร้าง UI Panel

ในขั้นตอนแรก เราจะออกแบบโครงสร้างอินเทอร์เฟซโดยแบ่งพื้นที่หน้าจอออกเป็นสามส่วนหลัก โดยใช้ ui.Panel เป็นองค์ประกอบหลัก พาเนลด้านซ้าย (layerPanel) ถูกกำหนดให้มีความกว้างคงที่ 300 พิกเซล พื้นหลังสีขาว และมีระยะเว้นขอบภายใน (padding) 8 พิกเซล ภายในพาเนลนี้เราจะวางป้ายชื่อ “Layers” เพื่อทำหน้าที่เป็นหัวข้อ กำหนดレイเอต์ให้เป็นแนวตั้ง (vertical flow) เพื่อให้ check box หรือคอนโทรลต่าง ๆ เรียงตัวกันลงมาในส่วนตรงกลาง เราสร้าง ui.Map ที่เป็นตัวแผ่นที่หลัก เปิดการควบคุมการซูมแผนที่ (zoomControl:true) พร้อมตั้งค่าให้แผนที่ขยายเต็มพื้นที่ (stretch:both) และกำหนดจุดกึ่งกลางและระดับซูมเริ่มต้น เพื่อให้ผู้ใช้เห็นข้อมูลทันทีเมื่อโหลด ในขณะที่พาเนลด้านขวา (chartPanel) ก็ถูกกำหนดลักษณะคล้ายกับพาเนลซ้าย แต่ใช้แสดงกราฟหรือการแสดงผลเชิงวิเคราะห์ โดยมีป้ายชื่อ “Chart” นำหน้า เมื่อเตรียมพาเนลทั้งสามเสร็จ เรานำมาต่อกันในพาเนลหลัก (root) ด้วยレイเอต์แนวนอน (horizontal flow) และสั่งให้ระบบล้างค่าเดิมจาก บี.รูท ก่อนจะเพิ่มพาเนลหลักเข้าไป ทำให้หน้าจอถูกจัดแบ่งเป็นสามคอลัมน์

source code:

```
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter7.md#71-create-ui-panel
// Create the left "layer" panel
var layerPanel = ui.Panel({
  layout: ui.Panel.Layout.flow('vertical'),
  style: {
    width: '300px',
    backgroundColor: '#fff',
    padding: '8px'
  }
});
layerPanel.add(ui.Label('Layers', {fontWeight: 'bold'}));

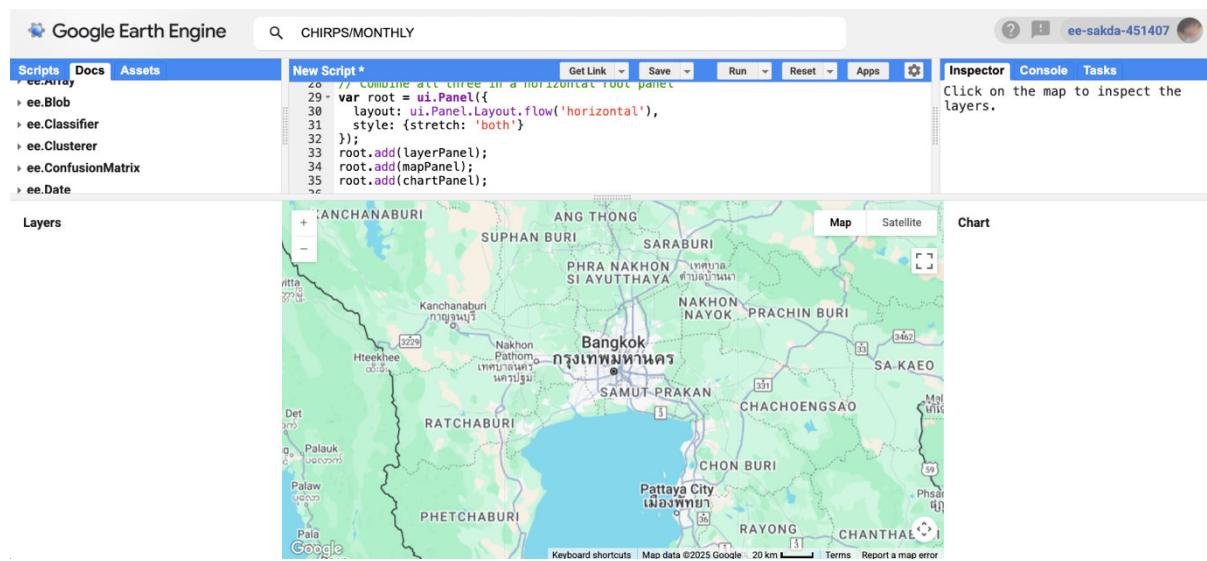
// Create the centre map panel
var mapPanel = ui.Map();
mapPanel.setControlVisibility({all: true, zoomControl: true});
mapPanel.style().set({stretch: 'both'}); // fill available space
mapPanel.setCenter(100.5, 13.7, 8);

// Create the right "chart" panel
var chartPanel = ui.Panel({
  layout: ui.Panel.Layout.flow('vertical'),
  style: {
    width: '300px',
    backgroundColor: '#fff',
    padding: '8px'
  }
});
chartPanel.add(ui.Label('Chart', {fontWeight: 'bold'}));

// Combine all three in a horizontal root panel
var root = ui.Panel({
  layout: ui.Panel.Layout.flow('horizontal'),
  style: {stretch: 'both'}
});
root.add(layerPanel);
root.add(mapPanel);
root.add(chartPanel);

// Render
ui.root.clear();
ui.root.add(root);
```

ผลลัพธ์การทำงานของโค้ด



7.4.2 เพิ่มขั้นข้อมูล

เมื่อโครงสร้าง UI พร้อมแล้ว เราจึงเตรียมข้อมูลเชิงพื้นที่และภาพดาวเทียมเพื่อนำมาแสดงเป็นขั้นข้อมูล ในที่นี้เริ่มจากการสร้าง ee.Geometry.Polygon เพื่อกำหนดพื้นที่สนใจ และแปลงขอบเขตเป็นเส้นขอบ (polyline) ด้วย .bounds() จากนั้นเขียนฟังก์ชันช่วยสองตัว ได้แก่ clipImage(image) สำหรับตัดภาพให้ครอบคลุมเฉพาะพื้นที่ polygon และ calculateNDVI(image) เพื่อคำนวณดัชนี NDVI และต่อแบบเดาไปในภาพเดิม ต่อมาก็ลดชุดภาพ Sentinel-2 ทั้งช่วงวันที่และพื้นที่ที่กำหนด ใช้ .map() เรียกทั้งสองฟังก์ชันตามลำดับ และคำนวณภาพ composite แบบ median ทุกแบบเดาเพื่อให้ได้ภาพที่ลดผลกระทบของเมฆและ noise หลังจากนั้นสร้างแบบเดา NDVI ใหม่โดยใช้วิธี normalizedDifference ระหว่างแบบเดา B8 และ B4 และรวมเข้ากับ composite เดิม

source code

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter7.md#72-add-layers

```
// Add layers
var polygon = ee.Geometry.Polygon(
  [[[98.9171009716561, 18.815619476862654],
    [98.9171009716561, 18.68557890893041],
    [99.0873890575936, 18.68557890893041],
    [99.0873890575936, 18.815619476862654]]]);
// convert polygon to polyline
var polyline = polygon.bounds().coordinates().get(0);

// Clip function
function clipImage(image) {
  return image.clip(polygon);
}
```

```

// Calculate NDVI function
function calculateNDVI(image) {
  var ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
}

// Create a polyline feature
var polylineFeature = ee.Feature(ee.Geometry.LineString(polyline), {name: 'Polygon Boundary'});

var s2 = ee.ImageCollection('COPERNICUS/S2')
  .filterDate('2025-02-01', '2025-06-28')
  .filterBounds(polygon)
  .map(clipImage)
  .map(calculateNDVI);

var composite = s2.median();

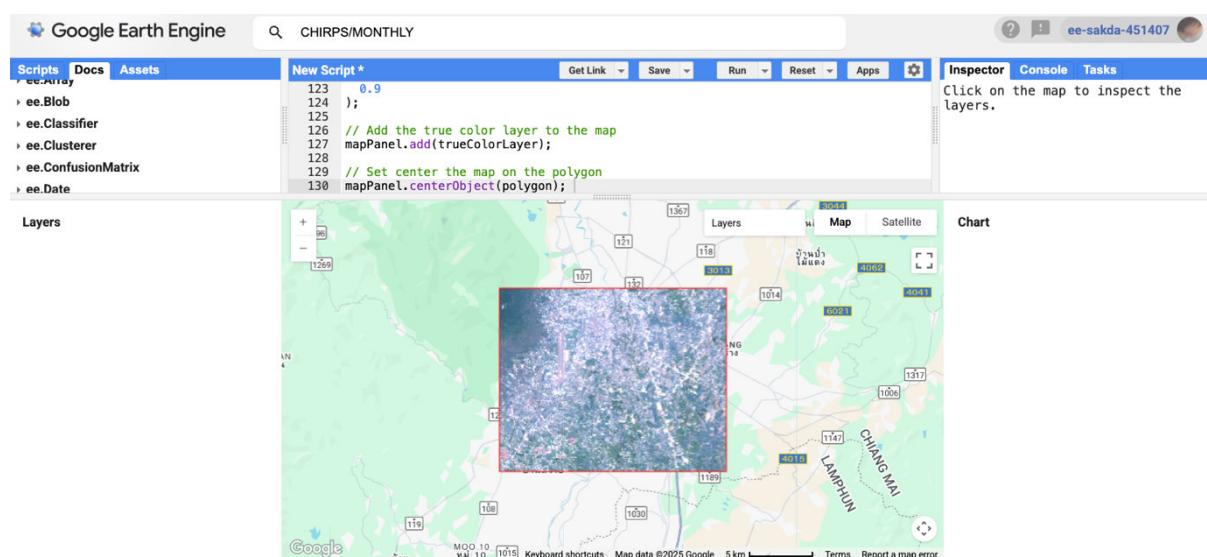
// Select bands for NDVI calculation
var nirBand = composite.select('B8'); // NIR band
var redBand = composite.select('B4'); // Red band

// Calculate NDVI
var ndvi = composite.normalizedDifference(['B8', 'B4']).rename('NDVI');

// Add NDVI to the composite
var compositeWithNDVI = composite.addBands(ndvi);

```

ผลลัพธ์การทำงานของโค้ด



จากนั้นกำหนดพารามิเตอร์การแสดงผลสำหรับภาพ NDVI (ค่า min/max และพาเลตต์สี) และสำหรับภาพจริง (true color) แล้วสร้าง ui.Map.Layer สำหรับแต่ละชั้นข้อมูล ได้แก่ ชั้นข้อมูล NDVI ชั้นข้อมูลเส้นขอบ

polygon และชั้นข้อมูล true color composite สุดท้ายจึงเพิ่มแต่ละชั้นข้อมูลเข้าไปใน mapPanel พร้อมตั้งศูนย์กลางແນວที่ให้ครอบคลุมพื้นที่ polygon

```
// Define visualization parameters for NDVI
var ndviVis = {
  min: -1,
  max: 1,
  palette: ['blue', 'white', 'green']
};

// Define visualization parameters for the composite
var compositeVis = {
  bands: ['B4', 'B3', 'B2'],
  min: 1000,
  max: 2700,
  gamma: 0.5
};

// Create an NDVI layer
var ndviLayer = ui.Map.Layer(
  compositeWithNDVI.select('NDVI'),
  ndviVis,
  'NDVI',
  true,
  0.9
);

// Add the NDVI layer to the map
mapPanel.add(ndviLayer);

// Add the polyline feature to the map
var polylineLayer = ui.Map.Layer(
  polylineFeature.geometry(),
  {color: 'red', width: 2},
  'Polygon Boundary'
);

// Add the polyline layer to the map
mapPanel.add(polylineLayer);

// Create a true color layer
var trueColorLayer = ui.Map.Layer(
  composite, compositeVis,
  'True Color Composite',
  true,
  0.9
);

// Add the true color layer to the map
mapPanel.add(trueColorLayer);
```

```
// Set center the map on the polygon
mapPanel.centerObject(polygon);
```

7.4.3 เพิ่ม Check box

เพื่อให้ผู้ใช้สามารถควบคุมการแสดงผลชั้นข้อมูลได้อย่างยืดหยุ่น เราจึงเพิ่ม ui.Checkbox ลงใน layerPanel สำหรับแต่ละชั้นข้อมูลที่สร้างไว้ โดยกำหนด label ให้สื่อความหมาย เช่น “NDVI”, “True Color” และ “Polygon Boundary” ตั้งค่าเริ่มต้น (value) เป็น true เพื่อให้ชั้นข้อมูลแสดงผลทันที และผูกเหตุการณ์ onChange เมื่อผู้ใช้คลิก check box จะเรียกใช้เมธอด mapPanel.layers().add() หรือ .remove() ตามสถานะของ check box ทำให้สามารถเปิดปิดชั้นข้อมูลแต่ละตัวได้ง่าย check box ทั้งหมดจะถูกเพิ่มเรียงกันใน layerPanel ทำให้ผู้ใช้เห็นปุ่มเลือกชั้นข้อมูลอยู่ทางด้านซ้ายของหน้าจออย่างชัดเจน

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter7.md#73-add-checkboxes

```
// Checkbox for NDVI
var ndviCheckbox = ui.Checkbox({
  label: 'NDVI',
  value: true,
  onChange: function(checked) {
    if (checked) {
      mapPanel.layers().add(ndviLayer);
    } else {
      mapPanel.layers().remove(ndviLayer);
    }
  }
});

// Add the checkbox to the layer panel
layerPanel.add(ndviCheckbox);

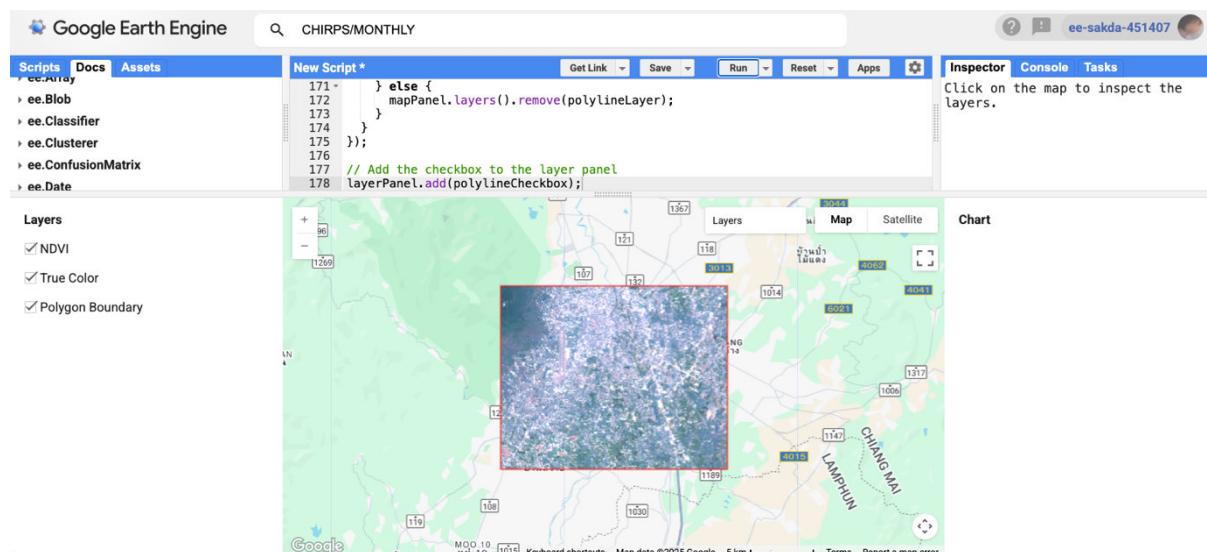
// Checkbox for True Color
var trueColorCheckbox = ui.Checkbox({
  label: 'True Color',
  value: true,
  onChange: function(checked) {
    if (checked) {
      mapPanel.layers().add(trueColorLayer);
    } else {
      mapPanel.layers().remove(trueColorLayer);
    }
  }
});

// Add the checkbox to the layer panel
layerPanel.add(trueColorCheckbox);
```

```
// Checkbox for Polygon Boundary
var polylineCheckbox = ui.Checkbox({
  label: 'Polygon Boundary',
  value: true,
  onChange: function(checked) {
    if (checked) {
      mapPanel.layers().add(polylineLayer);
    } else {
      mapPanel.layers().remove(polylineLayer);
    }
  }
});

// Add the checkbox to the layer panel
layerPanel.add(polylineCheckbox);
```

ผลลัพธ์การทำงานของโค้ด



7.4.4 เพิ่มปุ่มแสดงกราฟ

ในส่วนของพาเนลขวา (chartPanel) เราเพิ่มปุ่ม ui.Button สองปุ่มหลัก เพื่อเรียกดูกราฟ ปุ่มแรก “Show NDVI Chart” เมื่อผู้ใช้คลิกจะสร้างกราฟ histogram ด้วย ui.Chart.image.histogram โดยป้อนภาพ NDVI ความละเอียด 30 เมตร จากนั้นกำหนดตัวเลือกของกราฟ เช่น ชื่อเรื่อง แกน X แกน Y ความหนาของเส้น และขนาดจุด แล้วเพิ่มกราฟลงใน chartPanel ปุ่มที่สอง “Show NDVI Time Series” สร้างกราฟ time series ด้วย ui.Chart.image.series โดยใช้ชุดภาพ s2 ที่มี NDVI เป็นแบบดัชนี ระบุ region และ scale เช่นเดียวกับกราฟแรก พร้อมตั้งชื่อและสไตล์ กราฟที่ได้ถูกเติมลงใน chartPanel ทำให้ผู้ใช้สามารถคลิกปุ่ม เพื่อเรียกดูการแจกแจงค่า NDVI หรือการเปลี่ยนแปลง NDVI ตามเวลาได้ทันทีโดยไม่ต้องเขียนโค้ดใหม่ซ้ำ ในหน้าเดียวกัน

source code:

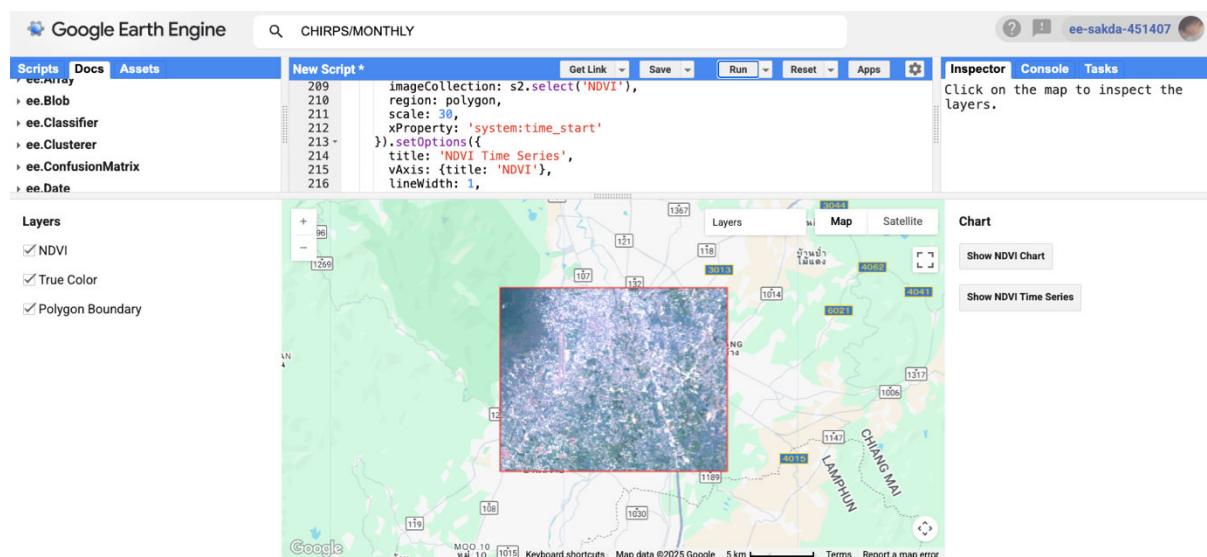
```
https://github.com/gistnorth/gistnorth\_gee/blob/main/gee\_workshop\_chapter7.md#74-add-chart-buttons
// Add a button to the chart panel
var chartButton = ui.Button({
  label: 'Show NDVI Chart',
  onClick: function() {
    // Create a chart for NDVI
    var chart = ui.Chart.image.histogram({
      image: compositeWithNDVI.select('NDVI'),
      region: polygon,
      scale: 30,
      minBucketWidth: 0.01
    }).setOptions({
      title: 'NDVI Histogram',
      hAxis: {title: 'NDVI'},
      vAxis: {title: 'Frequency'},
      lineWidth: 1,
      pointSize: 0
    });
    // Add the chart to the chart panel
    chartPanel.add(chart);
  }
});

// Add the button to the chart panel
chartPanel.add(chartButton);

var timeSeriesButton = ui.Button({
  label: 'Show NDVI Time Series',
  onClick: function() {
    // Create a time series chart for NDVI
    var timeSeriesChart = ui.Chart.image.series({
      imageCollection: s2.select('NDVI'),
      region: polygon,
      scale: 30,
      xProperty: 'system:time_start'
    }).setOptions({
      title: 'NDVI Time Series',
      vAxis: {title: 'NDVI'},
      lineWidth: 1,
      pointSize: 0
    });
    // Add the chart to the chart panel
    chartPanel.add(timeSeriesChart);
  }
});

// Add the time series button to the chart panel
chartPanel.add(timeSeriesButton);
```

ผลลัพธ์การทำงานของโค้ด



7.5 การเผยแพร่ GEE App

การเผยแพร่แอปพลิเคชันบนแพลตฟอร์ม Google Earth Engine (GEE App) เป็นกระบวนการสำคัญที่ช่วยให้ผู้ใช้ทั่วไปสามารถเข้าถึงแอปพลิเคชันที่เราออกแบบไว้โดยไม่ต้องเขียนโค้ดเอง การเผยแพร่มีขั้นตอนและแนวปฏิบัติหลัก ดังนี้

เริ่มจากพัฒนาสคริปต์ใน GEE Code Editor ให้ครบถ้วน ทั้งการโหลดข้อมูล การประมวลผล และองค์ประกอบ UI ที่ต้องการ เช่น แผนที่ ปุ่มควบคุม เช็คบ็อกซ์ และกราฟ หลังทดสอบจนมั่นใจว่าไม่มีข้อผิดพลาดใด ๆ แล้ว ให้เข้าไปที่เมนู “App” เพื่อเปิดใช้งานฟังก์ชันสร้างแอป

The figure shows the GEE Code Editor with a 'New Script *' tab. The code editor contains the following JavaScript code:

```

5   width: '300px',
6   backgroundColor: '#fff',
7   padding: '8px'
8 }
9 });
10 layerPanel.add(ui.Label('Layers', {fontWeight: 'bold'}));
11 // e.g. layerPanel.add(ui.Checkbox('NDVI', true));
12
13
14 // 2. Create the centre map panel
15 var mapPanel = ui.Map();
16 mapPanel.setControlVisibility({all: true, zoomControl: true});
17 mapPanel.style().set({stretch: 'both'}); // fill available space
18 mapPanel.setCenter(100.5, 13.7, 8);
19
20

```

The code defines a 'layerPanel' and a 'mapPanel'. The 'layerPanel' is a UI panel containing a bold 'Layers' label. The 'mapPanel' is a map component set to stretch to fit its container and centered at coordinates (100.5, 13.7) with a zoom level of 8.

จากนั้นคลิก NEW APP เพื่อเริ่มสร้าง Application

Manage Apps

[ADD CLOUD PROJECT](#)
[NEW APP](#)

x cloud/ee-sakda-451407

[SHARE PROJECT](#)
[VIEW GALLERY](#)

App Name (click to launch)

ID (click to update app) ↕

Delete

คลิกเลือก Cloud Project

Publish New App

Editing Access

Editing Access

Name and URL

Source Code

Publication and Viewers



Choose editing access

You won't be able to change this later

Cloud Project

Anyone you add with certain roles — including Owner, Editor, and Earth Engine Apps Publisher — can edit the app. [Learn more](#)

Project
ee-sakda-homhuan

Site: ee-sakda-homhuan.projects.earthengine.app

Only you

Only you can edit the app

CANCEL

NEXT

เมื่อคลิก Next ระบบจะขอให้กำหนดชื่อแอป (App Name) ซึ่งจะกลายเป็นส่วนหนึ่งของ URL เช่น

<https://earthengine.app/view/your-app-name>

Publish New App

Name and URL

Editing Access

Name and URL

Source Code

Publication and Viewers



App Name and URL

App Name
gistnorth-app

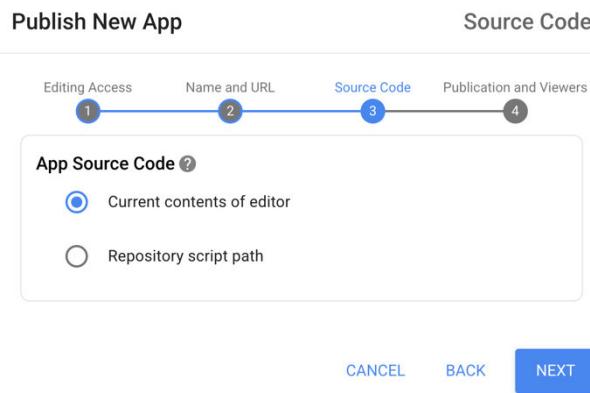
URL: <https://ee-sakda-homhuan.projects.earthengine.app/view/gistnor th-app>

CANCEL

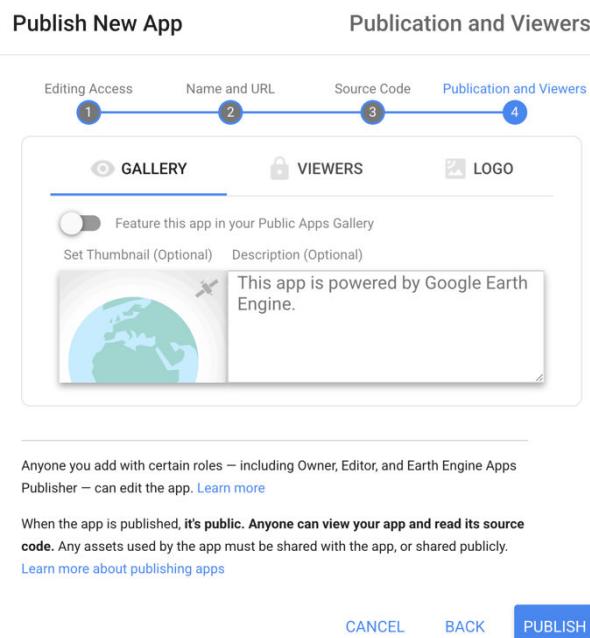
BACK

NEXT

เลือก source code ที่ต้องการจะ public โดยสามารถเลือกจากไฟล์ปัจจุบันหรือไฟล์อื่นๆ ที่เขียนไว้แล้ว



เมื่อมาถึงหน้าต่างอันสุดท้าย เราสามารถใส่โลโกของหน่วยงานของเรา

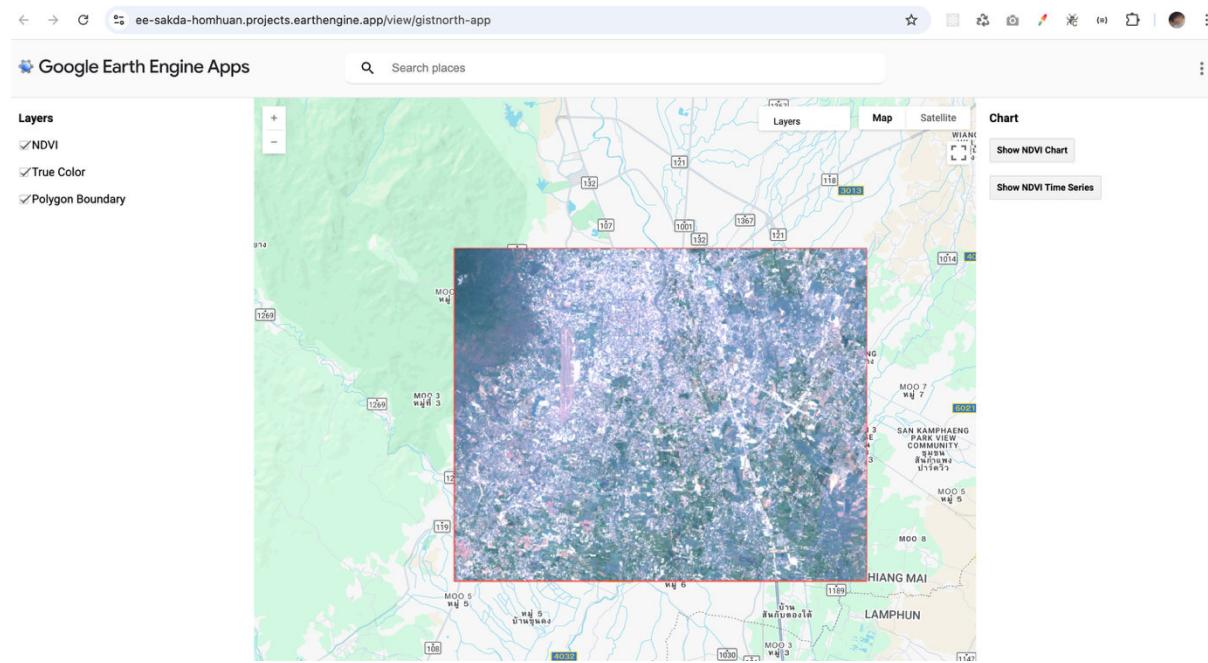


เมื่อคลิก Public สำเร็จจะปรากฏ รายการของ Application ใน cloud project ที่เราเลือก

× cloud/ee-sakda-homhuan		SHARE PROJECT	VIEW GALLERY
App Name (click to launch)	ID (click to update app)		Delete
biomass	cloud/ee-sakda-homhuan/biomass		
burnscar	cloud/ee-sakda-homhuan/burnscar		
flood	cloud/ee-sakda-homhuan/flood		
gistnorth-app	cloud/ee-sakda-homhuan/gistnorth-app		
ndwi-ndmi	cloud/ee-sakda-homhuan/ndwi-ndmi		
3pgs	cloud/ee-sakda-homhuan/pgs		

CLOSE

จากนั้นคลิกที่ชื่อ App เพื่อเปิด Application



หลังเผยแพร่แล้ว จะได้ลิงก์สั้น ๆ แบบเดียวกับบริการเว็บแอปทั่วไป เพียงส่งต่อให้ผู้สนใจ หรือฝังในหน้าเว็บ (Embedded iframe) เพื่อฝังอินเทอร์เฟซของแอปลงในเอกสาร HTML ของเว็บไซต์องค์กรหรือบล็อกส่วนตัว แต่ละครั้งที่เราแก้ไขสคริปต์ใน Code Editor จะต้องกด Publish ใหม่อีกครั้ง เพื่ออัปเดตเวอร์ชันบนเซิร์ฟเวอร์ ผู้ใช้ที่เข้าลิงก์เดิมจะเห็นการเปลี่ยนแปลงทันที แนะนำให้ดูบันทึกเวอร์ชันสำคัญ เช่น v1.0, v1.1 เพื่อการติดตามและแก้ไขปัญหาในอนาคต

สุดท้ายควรจัดทำคู่มือการใช้งานสั้น ๆ (Quick Start Guide) หรือวิดีโอสาธิตการใช้งาน เพื่อให้ผู้ใช้เข้าใจวิธีการควบคุม UI และตีความผลลัพธ์ได้อย่างถูกต้อง เช่น การเลือกวันเวลาชุดข้อมูล การเปิด–ปิดเลเยอร์ และการดาวน์โหลดผลลัพธ์

บทที่ 8 การจำแนกการใช้ประโยชน์ที่ดินและวิเคราะห์ความเปลี่ยนแปลง

การจำแนกการใช้ประโยชน์ที่ดินและการปักคุณพื้นดิน (Land Use & Land Cover Classification) ด้วย Google Earth Engine เป็นกระบวนการที่รวมการประมวลผลภาพดาวเทียม การสร้างชุดข้อมูลฝึกสอน (training data) และเทคนิคการเรียนรู้ภายใต้การดูแล (supervised learning) เข้าไว้ด้วยกัน ภายใต้โครงสร้างเชิร์ฟเวอร์ของ Earth Engine เราสามารถดึงข้อมูล Sentinel-2 ในช่วงเวลาต่างกัน สถาณลักษณะ สเปกตรัมที่สำคัญ ฝึกตัวจำแนก (classifier) และนำมาใช้จำแนกพื้นที่ในรูปแบบหลายชั้นได้อย่างรวดเร็ว บทความนี้จึงจะแบ่งออกเป็น 7 ส่วนหลัก เพื่ออธิบายตั้งแต่การเตรียมข้อมูล จนถึงการแสดงผลลัพธ์

8.1 กำหนดพื้นที่วิเคราะห์และโหลดข้อมูลฝึกสอน

ก่อนอื่นต้องกำหนดขอบเขตพื้นที่สนใจ (Region of Interest: ROI) ที่จะนำมาทำการจำแนก ซึ่งมักใช้พิกัดลองร่องในรูป Polygon เพื่อครอบคลุมพื้นที่เป้าหมายอย่างแม่นยำ จากนั้นเตรียมชุดข้อมูลฝึกสอนในรูปของ FeatureCollection ที่ประกอบด้วยจุดหรือพอลิกอนที่ติดป้ายรหัสคลาสใช้งาน เช่น น้ำ เมือง เกษตร ป่า เปเลือย รวมถึงข้อมูลคุณลักษณะ (properties) ซึ่ง landcover ตั้งแต่ 0-4 เพื่อใช้เป็นฐานข้อมูลนำร่องให้ตัวจำแนกเรียนรู้

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter8.md

```
// 1. Supervised land-use classification (5 classes) using Sentinel-2 at two time periods
var roi = ee.Geometry.Polygon([[[98.65287970599341, 17.722345177988142],
[98.65287970599341, 17.518162943397694],
[98.91105841693091, 17.518162943397694],
[98.91105841693091, 17.722345177988142]]]);
```



```
// 2. Load training data (FeatureCollection of points or polygons with a property 'landcover' 0-4)
// You must prepare this asset with 5 classes: e.g.
0=Water,1=Urban,2=Agriculture,3=Forest,4=Bare
var trainingFC = ee.FeatureCollection("projects/ee-sakda-451407/assets/trainning");

print(trainingFC);
```

8.2 การสร้างภาพ Composite จาก Sentinel-2 ในสองช่วงเวลา

เพื่อหลีกเลี่ยงปัญหาเมฆและ noise ของภาพเดียว เราจะดึงภาพ Sentinel-2_SR_HARMONIZED จาก Earth Engine ภายใน 2 ช่วงเวลาที่สนใจ — เช่น เดือนมกราคมในปีแรก และเดือนมกราคมในปีล่าสุด — และกรองภาพที่มีเมฆน้อยกว่า 20% ก่อนนำมาหา Median Composite ซึ่งจะสร้างภาพแทนค่ามรรคฐานของแต่ละแบบในแต่ละ ROI ผลลัพธ์คือภาพ composite ส่องชุดที่สะท้อนสภาพพื้นผิวในสองช่วงเวลา

```
// 3. Load Sentinel-2 and build composites for two date ranges
var start1 = '2019-01-01', end1 = '2019-01-31';
var start2 = '2025-01-01', end2 = '2025-01-31';

function makeComposite(start, end) {
  return ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
    .filterDate(start, end)
    .filterBounds(roi)
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
    .median()
    .clip(roi);
}

var comp1 = makeComposite(start1, end1);
var comp2 = makeComposite(start2, end2);
```

8.3 การเลือกแบบจำลองและสุมตัวอย่างข้อมูลผึกสอน

สำหรับงานจำแนกการใช้ประโยชน์ที่ดิน แบนด์หลักที่มักเลือก ได้แก่ Blue, Green, Red, Near-Infrared (NIR), SWIR1 และ SWIR2 (B2, B3, B4, B8, B11, B12) จากนั้นใช้เมธอด sampleRegions เพื่อสุมค่าพิกเซลของ composite ที่อยู่ภายใต้แนว point/polygon ใน trainingFC แต่ละจุด พร้อมดึงค่า landcover มาเป็นป้ายกำกับ ซึ่งจะได้ชุดตัวอย่างสองชุด (samples1, samples2) สำหรับแต่ละ composite

```
// 4. Select spectral bands for classification
var bands = ['B2','B3','B4','B8','B11','B12']; // Blue, Green, Red, NIR, SWIR1, SWIR2
```

8.4 การฝึกตัวจำแนกและจำแนกภาพ

เมื่อมีชุดตัวอย่างแล้ว จะใช้ Random Forest Classifier (เช่น รากต้นไม้ 100 ต้น) ฝึกแยกข้อมูลด้วย train() โดยระบุ inputProperties เป็นชื่อแบนด์ และ classProperty เป็น landcover หลังจากตัวจำแนกถูกสร้างเสร็จ จะนำไปใช้ classify() กับ composite แต่ละช่วงเวลา เพื่อให้ได้ภาพจำแนกชั้นการใช้ประโยชน์ที่ดินออกมาก

```
// 5. Sample the composites at the training points
var samples1 = comp1.select(bands).sampleRegions({
  collection: trainingFC,
  properties: ['landcover'],
  scale: 10,
  geometries: true
});

var samples2 = comp2.select(bands).sampleRegions({
  collection: trainingFC,
  properties: ['landcover'],
  scale: 10,
```

```

    geometries: true
});

// 6. Train classifiers (Random Forest) for each period
var classifier1 = ee.Classifier.smileRandomForest(100)
  .train({
    features: samples1,
    classProperty: 'landcover',
    inputProperties: bands
  });

var classifier2 = ee.Classifier.smileRandomForest(100)
  .train({
    features: samples2,
    classProperty: 'landcover',
    inputProperties: bands
  });

// 7. Classify the composites
var classified1 = comp1.select(bands).classify(classifier1);
var classified2 = comp2.select(bands).classify(classifier2);

```

8.5 การกำหนดพาlettes สีเพื่อการแสดงผล

เพื่อให้ผลลัพธ์เข้าใจง่าย เลือกพาlettes สีที่สื่อความหมายแก่แต่ละคลาส เช่น สีน้ำเงินสำหรับน้ำ สีแดงสำหรับเมือง สีเขียวสำหรับพื้นที่เกษตร สีเขียวเข้มสำหรับป่า สีเหลืองสำหรับพื้นที่เปลือย กำหนดช่วงค่า min–max ที่เหมาะสม (0–3 ในกรณี 4 คลาส) ทำให้แผนที่ผลลัพธ์ดูเป็นกลุ่มสีที่เข้าใจง่าย

```

// 8. Define a 4-class palette
var palette = [
  '0000FF', // 0 = Water (blue)
  'FF0000', // 1 = Urban (red)
  '00FF00', // 2 = Agriculture (green)
  '007F00', // 3 = Forest (dark green)
];

```

8.6 การแสดงผลลัพธ์บนแผนที่

นำ composite จริงและภาพจำแนกทั้งสองช่วงเวลาไปแสดงบนแผนที่ โดยตั้งจุดกึ่งกลางให้ครอบคลุม ROI ปรับชื่อเลyer และระดับความโปร่งแสงตามต้องการ สามารถเปิด–ปิด layer เพื่อเปรียบเทียบก่อนและหลังได้ทันทีใน Earth Engine Code Editor

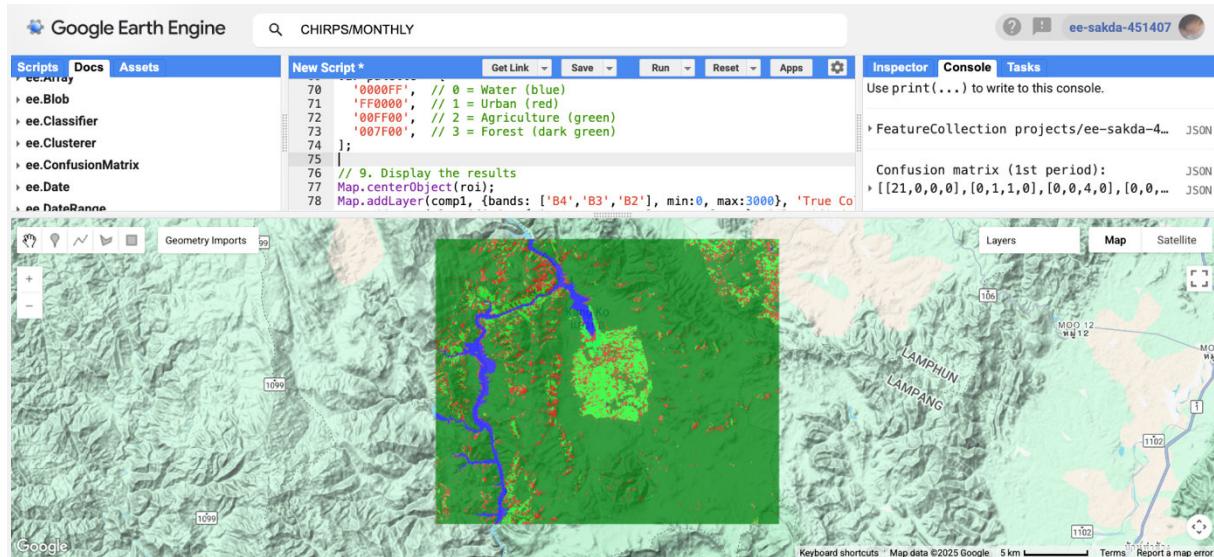
```

// 9. Display the results
Map.centerObject(roi);
Map.addLayer(comp1, {bands: ['B4', 'B3', 'B2'], min:0, max:3000}, 'True Color 1',
false);
Map.addLayer(classified1, {min:0, max:3, palette: palette}, 'Classified Jan-Mar',
true);

```

```
Map.addLayer(comp2, {bands: ['B4','B3','B2'], min:0, max:3000}, 'True Color 2', false);
Map.addLayer(classified2, {min:0, max:3, palette: palette}, 'Classified Jul-Sep', true);
```

ผลลัพธ์ที่ได้จากโค๊ด



8.7 การประเมินความแม่นยำ

เพื่อวัดประสิทธิภาพของตัวจำแนก แบ่งชุดตัวอย่างใน period แรกออกเป็นส่วนฝึก (70%) และทดสอบ (30%) แบบสุ่ม จากนั้นฝึก Random Forest อีกครั้งบนชุดฝึก และใช้ชุดทดสอบเรียก errorMatrix() คำนวณ confusion matrix พร้อม overall accuracy ซึ่งช่วยให้ประเมินจุดแข็ง-จุดอ่อนของโมเดลและปรับพารามิเตอร์หรือเพิ่มข้อมูลฝึกได้ต่อไป

```
// 10. Optional: accuracy assessment for period 1
var trainTest1 = samples1.randomColumn('rnd', 42);
var split = 0.7;
var trainSet = trainTest1.filter(ee.Filter.lt('rnd', split));
var testSet = trainTest1.filter(ee.Filter.gte('rnd', split));

var trainedRF = ee.Classifier.smileRandomForest(100)
    .train({features: trainSet, classProperty: 'landcover',
inputProperties: bands});

var validated = testSet.classify(trainedRF);

var testAccuracy = validated.errorMatrix('landcover', 'classification');
print('Confusion matrix (1st period):', testAccuracy);
print('Overall accuracy:', testAccuracy.accuracy());
```

confusion matrix ที่ได้สะท้อนภาพรวมของประสิทธิภาพการจำแนกคลาสทั้ง 4 การใช้ประโยชน์ที่ดิน ดังนี้

```

Inspector Console Tasks
Use print(...) to write to this console.

▶ FeatureCollection projects/ee-sakda-451407/assets/trainning (4 elements, 2 column... JSON

Confusion matrix (1st period):
[[21,0,0,0],[0,1,1,0],[0,0,4,0],[0,0,0,922]]
  ↗ 0: [21,0,0,0] JSON
  ↗ 1: [0,1,1,0] JSON
  ↗ 2: [0,0,4,0]
  ↗ 3: [0,0,0,922]

Overall accuracy: 0.9989462592202318 JSON

```

เมื่ออ่านจากແກ່ແກຣັງແທນຄໍາຈິງຂອງคลາສ Water จะเห็นວ່າມີເດລທໍານາຍທຸກຕ້ອງຢ່າງໄດ້ຖືກຕ້ອງທັງໝາດ 21 ຈຸດ ໄນມີຈຸດໃດທີ່ຜິດໄປເປັນຄລາສອື່ນເລີຍ ນັ້ນໜາຍຄວາມວ່າ ໂມີເດລເຮັດວຽກລັກຊະນະສຳຄັງຂອງພື້ນທີ່ນໍາໄດ້ຫັດເຈນແລະ ແມ່ນຢໍາ 100%

ສໍາຮັບຄລາສ Urban ງີ່ມີເພີ່ມ 2 ຈຸດໃນຊຸດທດສອບ ຄວາມລະເອີຍດີໃນການຈຳນັກແປປັນຄອນຂ່າງມາກ ເພຣະແມ່ ໂມີເດລຈະທໍານາຍໄດ້ຖືກຕ້ອງ 1 ຈຸດ ອີກ 1 ຈຸດກລັບຖືກເຂົ້າໃຈຜິດໄປເປັນຄລາສ Agriculture ແສດງໃຫ້ເຫັນວ່າລັກຊະນະທາງສປັກຕົວມະຫວາງເມືອງແລະພື້ນທີ່ເກະຕຣອາຈມີຄວາມໄກລ໌ເຄີຍກັນໃນກຣນິ້ນີ້ ຈຶ່ງເກີດກາຮັບສັນຮະຫວາງສອງຄລາສນີ້

ສ່ວນຄລາສ Agriculture ເອງ ແມ່ຈະມີຕ້ວອຍ່າງເພີ່ມ 4 ຈຸດ ແຕ່ໂມເດລກ໌ສາມາດຈຳນັກໄດ້ຮັບທຸກຈຸດອ່າຍ່າງສມູງຮຸນ ໄນມີການຮ່ວ້າໄລໄປຄລາສອື່ນ ຂະໜາທີ່ຄລາສ Forest ມີຕ້ວອຍ່າງຈຳນວນມາກທີ່ສຸດຕື້ນີ້ 922 ຈຸດ ແລະໂມເດລກ໌ຈັດກາໄດ້ໂດຍໄມ້ມີຈຸດໃດທຸກໆລັນ ນັ້ນສະຫຼອນວ່າປ່າໄມ້ໃນພື້ນທີ່ນີ້ມີລັກຊະນະສປັກຕົວມະເພາະຕ້ວໜັດເຈນ ທຳໄຫ້ໂມເດລຈະຈຳໄດ້ແມ່ນຢໍາ

บทที่ 9 การติดตามสถานการณ์ภัยพิบัติ

9.1 การวิเคราะห์พื้นที่น้ำท่วม

การตรวจจับพื้นที่น้ำท่วมด้วยข้อมูล Sentinel-1 SAR บนแพลตฟอร์ม Google Earth Engine เป็นกระบวนการสำคัญในการจัดทำแผนที่น้ำท่วมอย่างรวดเร็วและแม่นยำ ข้อมูล SAR (Synthetic Aperture Radar) สามารถเจาะผ่านเมฆและใช้งานได้ทั้งกลางวันและกลางคืน จึงเหมาะสมสำหรับติดตามสถานการณ์น้ำท่วมที่มักเกิดพร้อมกับฝนตกหนักที่มาพร้อมกับเมฆปุกคลุมสูง ทั้งกระบวนการอ้างอิงแนวทางตามข้อแนะนำของ UN-SPIDER

9.1.1 กำหนดพื้นที่สนใจ

เพื่อให้การวิเคราะห์ครอบคลุมบริเวณที่ต้องการ เราจะกำหนดขอบเขตทางภูมิศาสตร์ (Region of Interest: ROI) ด้วยพิกัดในรูปแบบพอลิกอน จากนั้นแปลงเป็น FeatureCollection เพื่อให้ Earth Engine จัดการเป็นชุดข้อมูลเชิงพื้นที่ได้สะดวก การกำหนดพื้นที่ล่วงหน้าช่วยให้การคัดกรองภาพ Sentinel-1 มุ่งเน้นเฉพาะพิกเซลภายในขอบเขตนี้ ลดเวลาและทรัพยากรในการประมวลผล

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter9.md#define-the-region-of-interest

```
// 1. Define region of interest (adjust coordinates as needed)
var geometry = ee.Geometry.Polygon(
  [[[99.78173820585737, 20.530460641212706],
    [99.78173820585737, 20.343223656107035],
    [100.15664665312299, 20.343223656107035],
    [100.15664665312299, 20.530460641212706]]]);

// 2. Define the area of interest as a FeatureCollection
var aoi = ee.FeatureCollection(geometry);
```

9.1.2 โหลดข้อมูลก่อนและหลังเหตุการณ์น้ำท่วม

ขั้นตอนต่อมาเป็นการเลือกช่วงเวลาที่เหมาะสม แบ่งเป็นก่อนน้ำท่วม (ก่อนเหตุการณ์) และหลังน้ำท่วม (หลังเหตุการณ์) ซึ่งควรอ้างอิงวันที่เกิดน้ำท่วมจริงในพื้นที่นั้น ๆ การกำหนดช่วงเวลาชัดเจนช่วยให้วัดความเปลี่ยนแปลงของค่าการสะท้อนสัญญาณ (backscatter) ระหว่างสองช่วงเวลาได้แม่นยำยิ่งขึ้น

```
// 3. Define date ranges for before and after flood events
var before_start = '2024-01-01';
var before_end = '2024-05-24';
var after_start = '2024-09-15';
var after_end = '2024-10-10';
```

9.1.3 เลือกและกรองข้อมูล Sentinel-1 SAR

เมื่อได้ช่วงวันแล้ว จะโหลดชุดภาพ Sentinel-1 GRD (Ground Range Detected) บนโหมดอินเตอร์เฟอร์โรมิเตอร์ (IW) พร้อม polarization ที่สนใจ (VV หรือ VH) และทิศทางการบิน (ASCENDING หรือ DESCENDING) นอกจากนี้กรองภาพที่มีความละเอียด 10 เมตร และครอบทับ ROI เท่านั้น เพื่อให้มั่นใจว่าได้ข้อมูลที่สอดคล้องทั้งเชิงพื้นที่และเชิงเวลาที่ใช้เคราะห์

```
// 4. Define parameters for Sentinel-1 data
var polarization = "VH"; // 'VV' 'VH'
var pass_direction = "DESCENDING"; // 'DESCENDING' หรือ 'ASCENDING'
// Load and filter Sentinel-1 GRD
var collection = ee.ImageCollection('COPERNICUS/S1_GRD')
  .filter(ee.Filter.eq('instrumentMode', 'IW'))
  .filter(ee.Filter.listContains('transmitterReceiverPolarisation',
polarization))
  .filter(ee.Filter.eq('orbitProperties_pass', pass_direction))
  .filter(ee.Filter.eq('resolution_meters', 10))
//.filter(ee.Filter.eq('relativeOrbitNumber_start',relative_orbit ))
  .filterBounds(aoi)
  .select(polarization);
// Filter date
var before_collection = collection.filterDate(before_start, before_end);
var after_collection = collection.filterDate(after_start, after_end);

// Create mosaics for before and after periods
var before = before_collection.mosaic().clip(aoi);
var after = after_collection.mosaic().clip(aoi);
```

9.1.4 คำนวณความแตกต่างของ backscatter

หลังสร้างภาพ mosaic ของช่วงก่อนและหลังน้ำท่วม จะทำการกรองเชิงเนื้อที่ (focal mean) เพื่อเนียนค่าพิกเซลรอบ ๆ ตามรัศมีที่กำหนด ก่อนแปลงเป็นหน่วยเดซิเบล (dB) และนำค่าหลังลบค่าก่อนน้ำท่วม เพื่อให้ได้ภาพความแตกต่างของสัญญาณ หากค่าน้อยกว่าเกณฑ์ที่ตั้งไว้ จะถือว่าเป็นพื้นที่น้ำท่วมเบื้องตน (raw flood mask)

```
// 5. Calculate the difference in backscatter between the two periods
var smoothing_radius = 25;
var before_filtered = before.focal_mean(smoothing_radius, 'circle', 'meters');
var after_filtered = after.focal_mean(smoothing_radius, 'circle', 'meters');
// Define a threshold for flood detection
var difference_threshold = -5.5;
var difference_db = after_filtered.subtract(before_filtered);
var difference_binary = difference_db.lte(difference_threshold);
var flood_raw_mask = difference_db.updateMask(difference_binary);
```

9.1.5 ปรับปรุงน้ำท่วมด้วยเกณฑ์อื่นๆ

สิ่งที่เราเคราะห์ได้อาจเป็นพื้นที่แหล่งน้ำเดิม (permanent water) หรือจุดที่มีเสียงรบกวนอื่น ๆ จึงนำข้อมูล JRC Global Surface Water มาใช้ตรวจสอบ seasonality ของน้ำ และตั้งเงื่อนไขให้พื้นที่ที่เป็นน้ำอยู่เดิม (มีน้ำอยู่แล้ว > 5 เดือน) ไม่ถูกนับเป็นน้ำท่วม จากนั้นกรองจุดเดียวหรือกลุ่มเล็ก ๆ ด้วย connected pixel count เพื่อกำจัด noise สุดท้ายนำข้อมูลภูมิประเทศ (DEM) มาคำนวณ slope และตัดพื้นที่ลาดชันเกินเกณฑ์ เพื่อให้มาสก์น้ำท่วมโอบล้อมเฉพาะบริเวณที่เหมาะสม

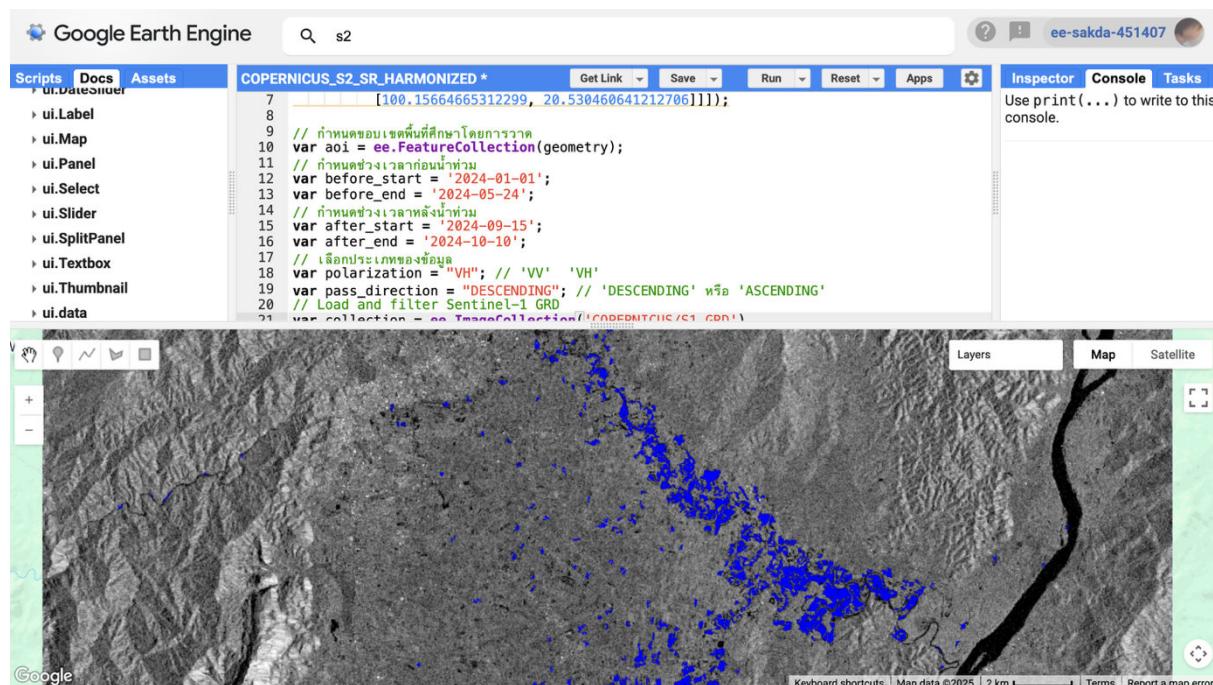
```
// 6. Refine the flood mask using additional criteria
var swater = ee.Image('JRC/GSW1_0/GlobalSurfaceWater').select('seasonality');
var swater_mask = swater.gte(5).updateMask(swater.gte(5));
var flooded_mask = difference_binary.where(swater_mask, 0);
var flooded = flooded_mask.updateMask(flooded_mask);
var connections = flooded.connectedPixelCount();
var flooded = flooded.updateMask(connections.gte(8));
var dem = ee.Image('WWF/HydroSHEDS/03VFDEM');
var terrain = ee.Algorithms.Terrain(dem);
var slope = terrain.select('slope');
var flooded = flooded.updateMask(slope.lt(5));
```

9.1.6 แสดงผลลัพธ์บนแผนที่

เมื่อได้น้ำท่วมสุดท้ายแล้ว จะนำภาพก่อน-หลัง ภาพความแตกต่าง และพื้นที่น้ำท่วมที่ปรับปรุงแล้ว มาแสดงบน Map ด้วยพาเลต์สีที่เหมาะสม ผู้ใช้สามารถเปิด-ปิดแต่ละレイเยอร์ เพื่อเปรียบเทียบสถานะพื้นที่ก่อนและหลังเหตุการณ์น้ำท่วมได้อย่างชัดเจน

```
// 7. Display the results
Map.centerObject(aoi);
Map.addLayer(before_filtered, { min: -25, max: 0 }, 'Before Flood', 0);
Map.addLayer(after_filtered, { min: -25, max: 0 }, 'After Flood', 1);
Map.addLayer(difference_db, { min: -5, max: 5 }, 'Difference (dB)', 0);
Map.addLayer(flood_raw_mask, { palette: 'blue' }, 'Flooded (raw)', 0);
Map.addLayer(flooded, { palette: 'blue' }, 'Flooded Areas', 1);
```

ผลลัพธ์ที่ได้จากโค้ด



9.2 การวิเคราะห์พื้นที่เลี่ยงภัยแล้ง

การติดตามภาวะภัยแล้ง (Drought Monitoring) เป็นหัวใจสำคัญในการวางแผนบริหารจัดการทรัพยากรน้ำ และบรรเทาผลกระทบต่อภาคเกษตรกรรมและชุมชน ในยุคที่ข้อมูลดาวเทียมพร้อมใช้งานผ่านแพลตฟอร์มคลาวด์ Google Earth Engine (GEE) การคำนวณดัชนี SPI (Standardized Precipitation Index) บนฐานข้อมูลปริมาณน้ำฝน CHIRPS (Climate Hazards Group InfraRed Precipitation with Station data) จึงกลายเป็นวิธีที่รวดเร็ว แม่นยำ และครอบคลุมพื้นที่กว้างใหญ่ เหมาะกับการตรวจสอบแนวโน้มภาวะแล้งในระดับภูมิภาคหรือระดับประเทศ

9.2.1 การโหลดขอบเขตประเทศไทยและข้อมูล CHIRPS

เริ่มต้นด้วยการกำหนดขอบเขตพื้นที่ศึกษาโดยนำพรมแดนประเทศไทยจากชุดข้อมูล FeatureCollection ของ US Department of State มาใช้เป็น Region of Interest จากนั้นโหลดชุดข้อมูล CHIRPS ประจำวัน (Daily) ซึ่งให้ค่าปริมาณน้ำฝนในแต่ละวันทั่วโลก การเลือกใช้ CHIRPS Daily ทำให้เราสามารถสรุปข้อมูลเป็นช่วงเวลาที่ต้องการได้ยืดหยุ่น ทั้งรายเดือน รายไตรมาส หรือรายปี

source code:

https://github.com/gistnorth/gistnorth_gee/blob/main/gee_workshop_chapter9.md#drought-monitoring-using-chirps-precipitation-data

```
// 1. Load Thailand boundary
var thailand = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017')
    .filter(ee.Filter.eq('country_na', 'Thailand'));
```

```
// 2. Load CHIRPS monthly precipitation and select the 'precipitation' band
var precip = ee.ImageCollection('UCSB-CHG/CHIRPS/DAILY')
    .select('precipitation');
```

9.2.2 การคำนวณ SPI สำหรับช่วงเวลา 3 เดือนปัจจุบัน

เพื่อประเมินภาวะแล้งในช่วงเวลาล่าสุด เรากำหนดวันสิ้นสุดเป็นเดือนเมษายน 2025 และย้อนหลังไปรวมปริมาณน้ำฝน 3 เดือนก่อนหน้า (กุมภาพันธ์–เมษายน) การสรุปค่าปริมาณน้ำฝนในไตรมาสล่าสุดช่วยให้เราเข้าใจว่าภาวะน้ำฝนในปัจจุบันเปรียบเทียบกับมาตรฐานระยะสั้นเป็นอย่างไร

```
// 3. Define the "current" 3-month period ending April 2025
var targetDate = ee.Date('2025-01-01');
var startPeriod = targetDate.advance(-3, 'month');
var current3mo = precip
    .filterDate(startPeriod, targetDate)
    .sum();
```

9.2.3 การสร้างชุดเวลาของผลรวมปริมาณน้ำฝนย้อนหลังช่วง 3 เดือน

เพื่อใช้เป็นฐานข้อมูลเปรียบเทียบ เราสร้างชุดเวลาอย้อนหลังตั้งแต่ปี 1981 ถึง 2024 ของผลรวมปริมาณน้ำฝนในช่วงเดียวกัน (ไตรมาสเมษายน–มิถุนายน) โดยวนลูปสร้างภาพรวมปริมาณน้ำฝน 3 เดือนสำหรับแต่ละปี การรวมชุดภาพเหล่านี้เป็น ImageCollection ทางประวัติศาสตร์ ทำให้เราสามารถคำนวณสถิติภาพรวมขึ้มมาได้

```
// 4. Build a time series of historical 3-month sums (April–June each year)
var years = ee.List.sequence(1981, 2024);
var historical3mo = ee.ImageCollection.fromImages(
  years.map(function(y) {
    var start = ee.Date.fromYMD(y, 1, 1).advance(3, 'month'); // window start = April 1
    var end   = start.advance(3, 'month');
    return precip
      .filterDate(start, end)
      .sum()
      .set('system:time_start', start.millis());
  })
);
```

9.2.4 การคำนวณค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานจากข้อมูลย้อนหลัง

เมื่อได้ ImageCollection ของผลกระทบปริมาณน้ำฝนย้อนหลังแล้ว ขั้นตอนต่อไปคือคำนวณภาพเฉลี่ย (mean) และภาพส่วนเบี่ยงเบนมาตรฐาน (standard deviation) บนเซอร์ฟเวอร์ GEE ผลลัพธ์ทั้งสองภาพนี้จะใช้เป็นค่ามาตรฐานในการเปรียบเทียบภาวะของไตรมาสปัจจุบัน

```
// 5. Compute mean and standard deviation images from the historical period
var mean3mo = historical3mo.mean();
var stddev3mo = historical3mo.reduce(ee.Reducer.stdDev());
```

9.2.5 การคำนวณค่า SPI-3 (ดัชนีความแปรปรวนมาตรฐาน)

ดัชนี SPI-3 คำนวณจากค่า standardized anomaly ซึ่งได้จากการนำค่าปริมาณน้ำฝนไตรมาสสามมาลบด้วยค่าเฉลี่ยย้อนหลัง และหารด้วยส่วนเบี่ยงเบนมาตรฐาน ผลลัพธ์คือภาพที่แสดงว่าแต่ละพื้นที่เปลี่ยนแปลงจากสภาพอากาศปกตินาดีหน่อยติดลบหมายถึงภาวะแล้ง และค่าสูงกว่าศูนย์หมายถึงพื้นที่ชุมชนกว่าปกติ

```
// 6. Calculate the SPI-3 (approximate) as standardized anomaly
var spi3 = current3mo
  .subtract(mean3mo)
  .divide(stddev3mo)
  .clip(thailand);
```

9.2.6 การตั้งค่าการแสดงผลของ SPI

เพื่อให้การแปลความหมายของแผนที่ชัดเจน เราจึงกำหนดช่วงค่าตั้งแต่ -2 ถึง +2 โดยใช้พาราเลลล์เส้นตั้งแต่เส้นเดenze (แดงรุนแรง) สีเขียว (เข้ม) เหลือง จนถึงสีเขียว (เข้ม) ทั้งนี้การเลือกเกณฑ์สีและช่วงค่าแนะนำให้สอดคล้องกับมาตรฐานสากลของ SPI

```
// 7. Visualization parameters for SPI
var spiVis = {
  min: -2,
  max: 2,
  palette: [
    '#d73027', // <= -1.5 (severe drought)
    '#fc8d59', // -1.5 to -1.0
    '#fee08b', // -1.0 to -0.5
    '#d9ef8b', // -0.5 to 0.5 (near normal)
    '#91cf60', // 0.5 to 1.0
    '#1a9850' // > 1.0 (wet)
  ]
};
```

9.2.7 การแสดงผลแผนที่ SPI พร้อมสัญลักษณ์คำอธิบาย (Legend)

เมื่อได้ภาพ SPI-3 สำหรับติดมาสู่สุดแล้ว นำมาแสดงบน Map โดยศูนย์กลางที่ประเทศไทย พร้อมเพิ่มเลเยอร์และແຜງคำอธิบายສີ (legend) ປຶ້ງປະກອບດ້ວຍສัญลักษນໍສີແລະໜ່າງຄ່າ ໜ່າຍໃຫ້ຜູ້ໃຊ້ຈາກຮຸນແຮງຂອງກາວະແລງໄດ້ຍ່າງຮວດເຮົາ

```
// 8. Display the SPI layer
Map.centerObject(thailand, 6);
Map.addLayer(spi3, spiVis, 'SPI-3 Apr 2025');

// 9. Add a legend panel
var legend = ui.Panel({
  style: {
    position: 'bottom-left',
    padding: '8px 15px',
    backgroundColor: 'white',
    fontWeight: 'bold'
  }
});
legend.add(ui.Label('SPI-3 Legend'));

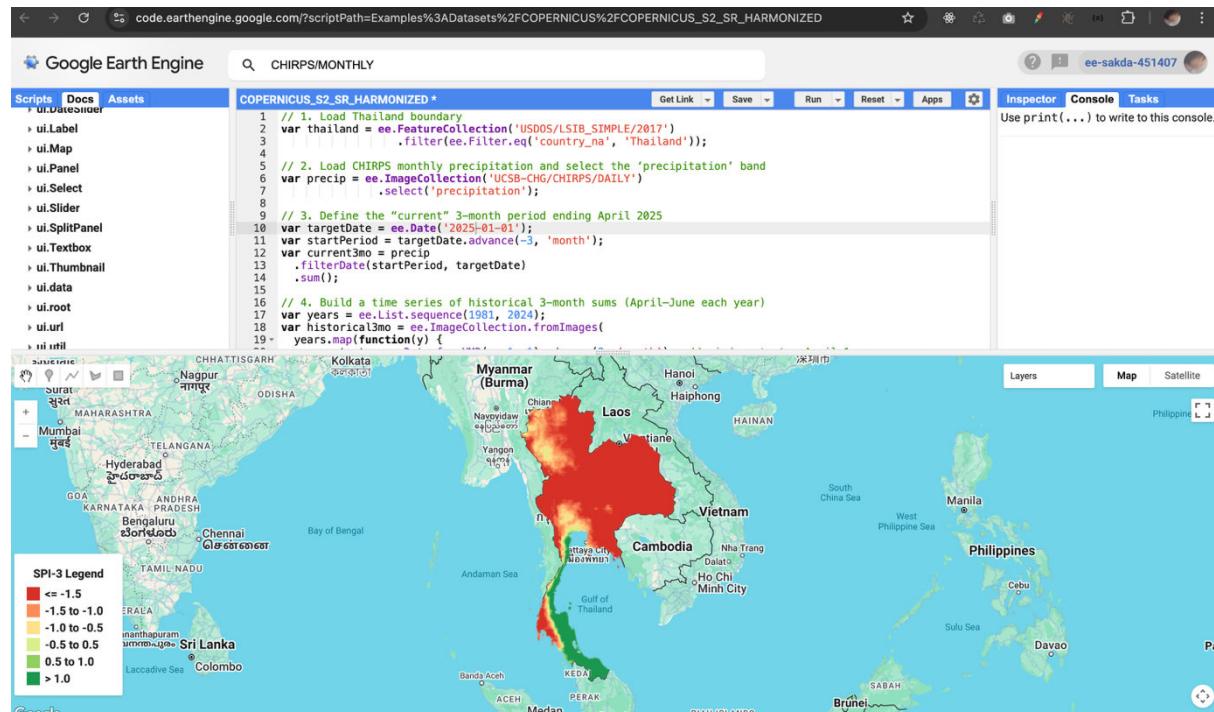
var makeRow = function(color, name) {
  var colorBox = ui.Label({
    style: {
      backgroundColor: color,
      padding: '8px',
      margin: '0 0 4px 0'
    }
  });
  var description = ui.Label(name, {margin: '0 0 4px 6px'});
  return ui.Panel([colorBox, description], ui.Panel.Layout.Flow('horizontal'));
};

var palette = spiVis.palette;
var names = ['<= -1.5', '-1.5 to -1.0', '-1.0 to -0.5',
             '-0.5 to 0.5', '0.5 to 1.0', '> 1.0'];

palette.forEach(function(color, i) {
  legend.add(makeRow(color, names[i]));
});

Map.add(legend);
```

ผลลัพธ์จากการโคด



เอกสารอ้างอิง

- European Commission, Joint Research Centre, United Nations Environment Programme, & Google. (2020). Global Surface Water Explorer [Web application]. สืบค้นเมื่อ 27 พฤษภาคม 2025, จาก <https://global-surface-water.appspot.com/map>
- Global Forest Watch. (n.d.). Global Forest Watch interactive map [Web application]. สืบค้นเมื่อ 27 พฤษภาคม 2025, จาก <https://www.globalforestwatch.org/map/>
- Google. (n.d.). Google Earth Engine. Retrieved May 27, 2025, from <https://earthengine.google.com>
- Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 202, 18–27. <https://doi.org/10.1016/j.rse.2017.06.031>
- Moore, R. (2010, December 2). Introducing Google Earth Engine [Blog post]. Google Official Blog. Retrieved from <http://blog.google.org/2010/12/introducing-google-earth-engine.html>
- SERVIR-SEA. (n.d.). Regional Land Cover Monitoring System: Land Cover [Web application]. สืบค้นเมื่อ 27 พฤษภาคม 2025, จาก <https://landcovermapping.org/en/landcover/>