GIS for Economists 6

Giorgio Chiovelli Sebastian Hohmann Tanner Regan

30/05/2022

Table of contents

Overview

The plan for today

Minimal introduction to remote sensing

- The electromagnetic spectrum
- Band combinations
- NDVI

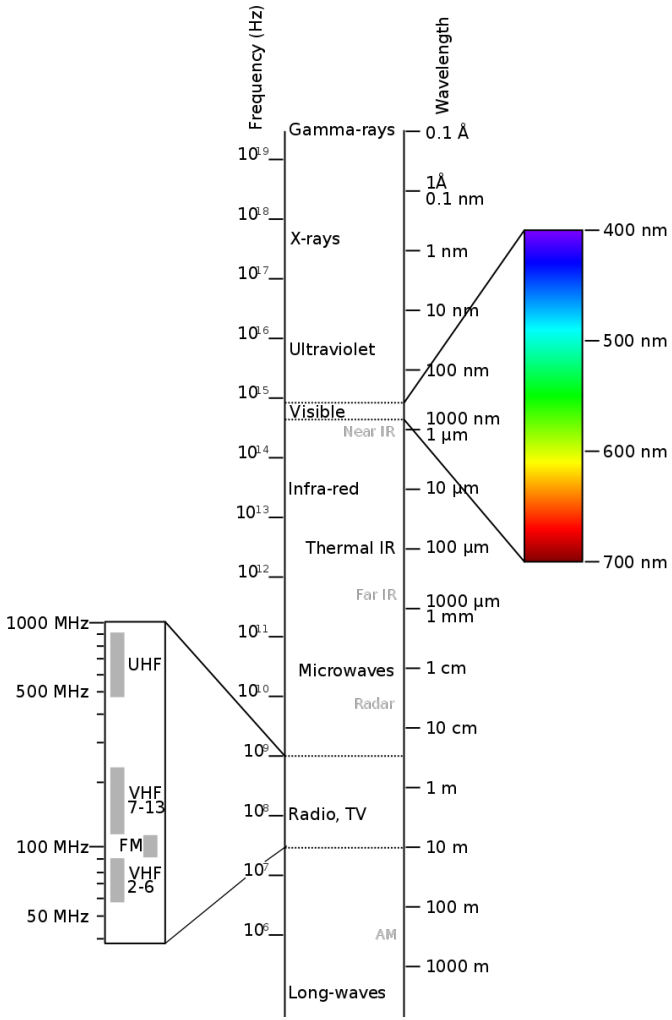
Introduction Google Earth Engine

- What is it?
- The GEE interface
- Finding and loading data
- Displaying data
- Basic calculations
- Exporting results
- Land cover classification with supervised learning

Remote sensing

The electromagnetic spectrum

The material in this section is taken from `https://www.youtube.com/watch?v=zesEJouvNU4`

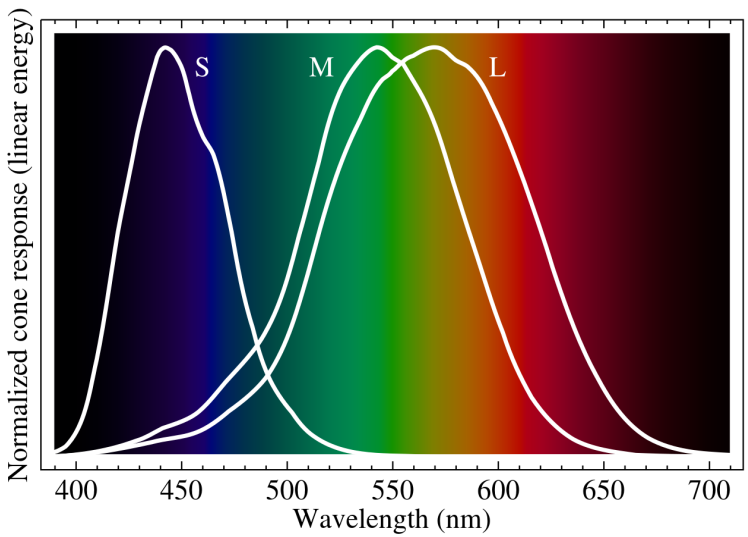


source: `https://en.wikipedia.org/wiki/Electromagnetic_spectrum`

- Energy from the sun constantly hits the earth as electromagnetic waves
  - Waves are categorized into a spectrum from very long frequency, low energy radio-waves to high frequency, high energy gamma waves
  - Only a tiny portion of the spectrum is visible to humans
  - To see images rather than just white brightness, human eyes and brains must differentiate and interpret energy in the visible portion of the spectrum

Remote sensing

Human visision

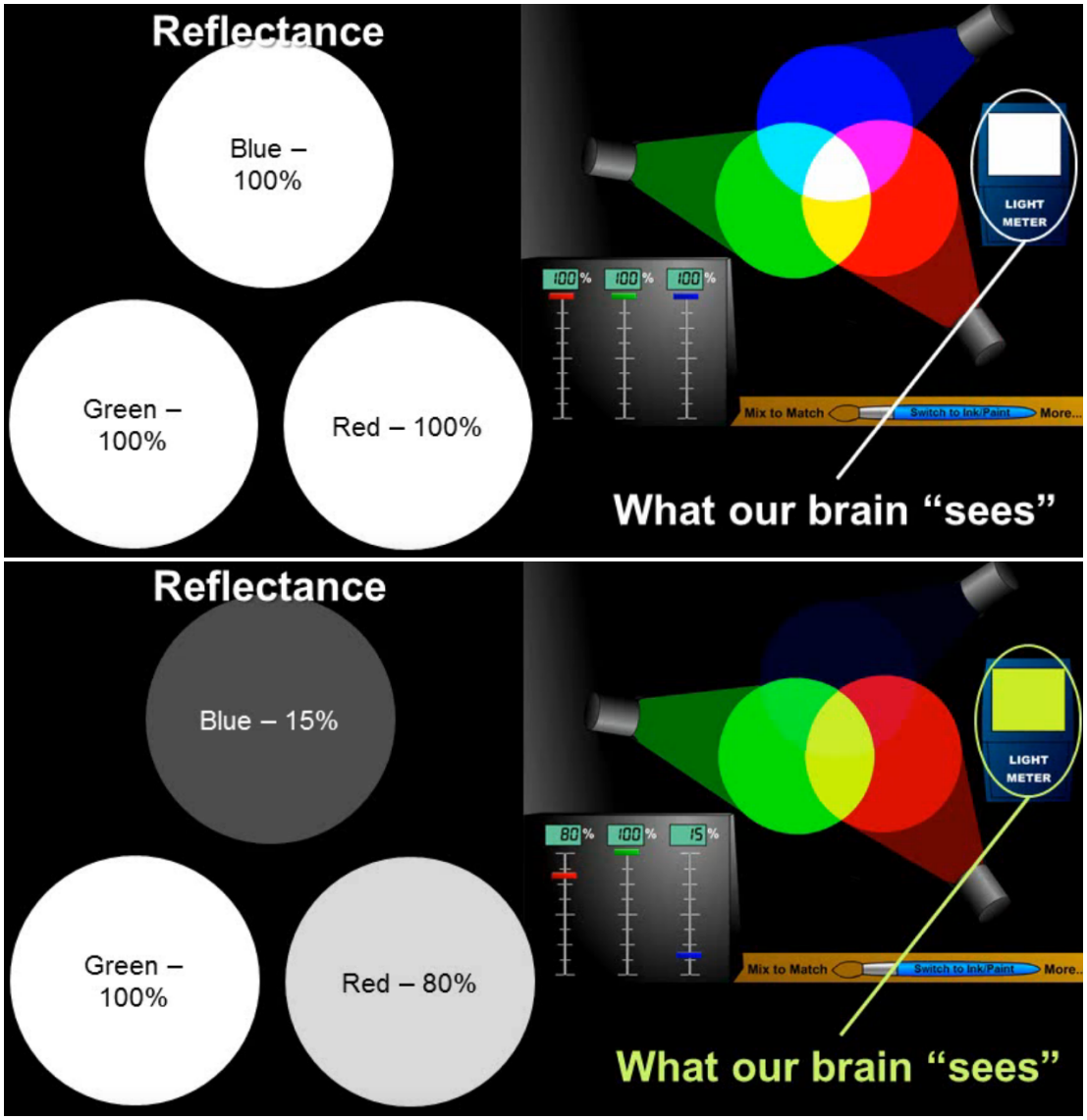


source: `https://en.wikipedia.org/wiki/Color_vision`

- We perceive color with specialized cone cells in the retina sensitive to different parts of the spectrum
  - There are three types, ordered by the parts of the spectrum at which their sensitivity peaks: short (S), medium (M), and long (L) cones
  - These wavelengths do not directly correspond to colors
  - Rather, the output of these cells gets processed in the brain to produce the final color images

Remote sensing

Human vision

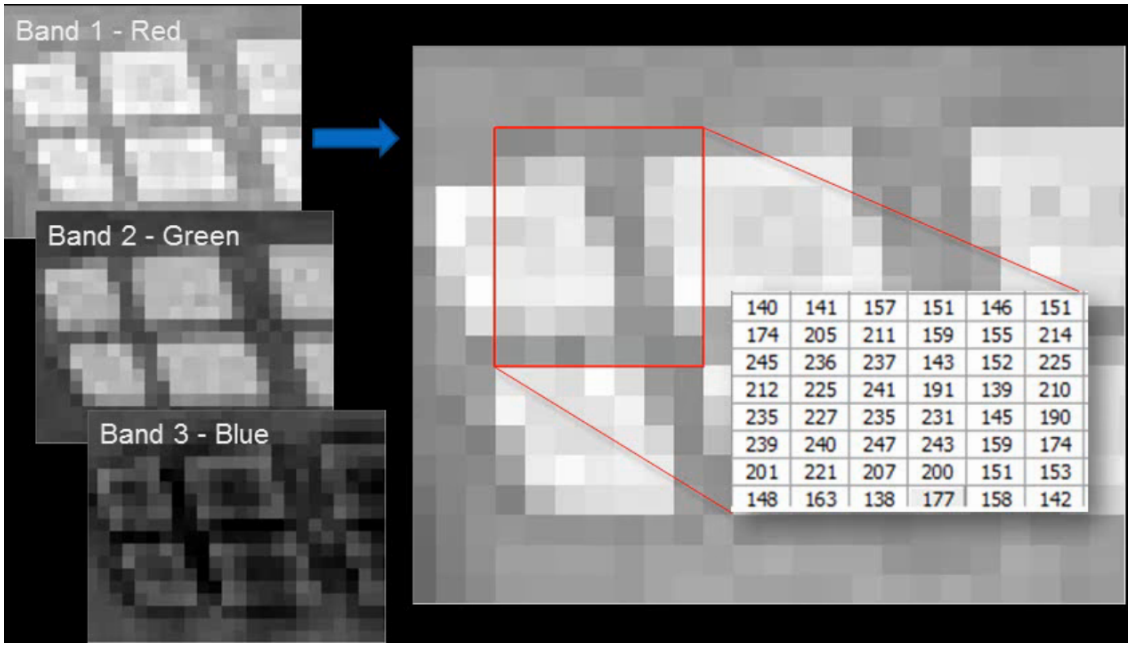


source: https://www.youtube.com/watch?v=zesEJouvNU4

- Combining reflectance at different intensities from different parts of the spectrum results in color vision
- Maximum reflectance from red, green, and blue band $\rightarrow$ perceive "white"
- Change relative intensity of reflectance $\rightarrow$ color perception changes to "yellow"

Remote sensing

Digital cameras



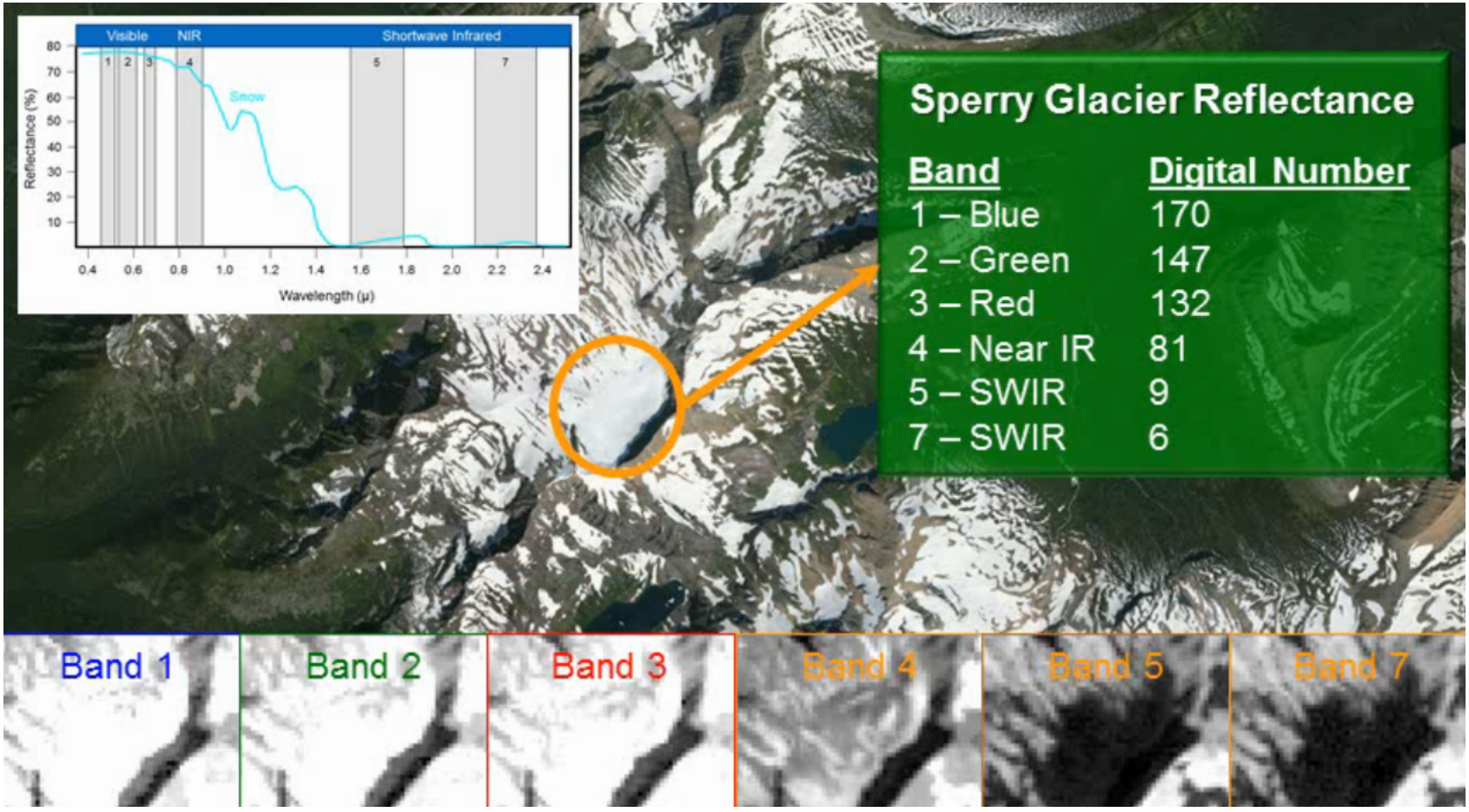
source: `https://www.youtube.com/watch?v=zesEJouvNU4`

- Most digital cameras record reflectance values for each pixel the three bands in digital numbers (DN) from 0 to 255
- Landsat 1-7 satellites also use this 8-bit recording; Landsat 8 satellites store 12-bit images (0 to 4095)
- Digital cameras then assign colors to the reflectance within each band and overlay the three bands to produce the final color image

Remote sensing

Satellites

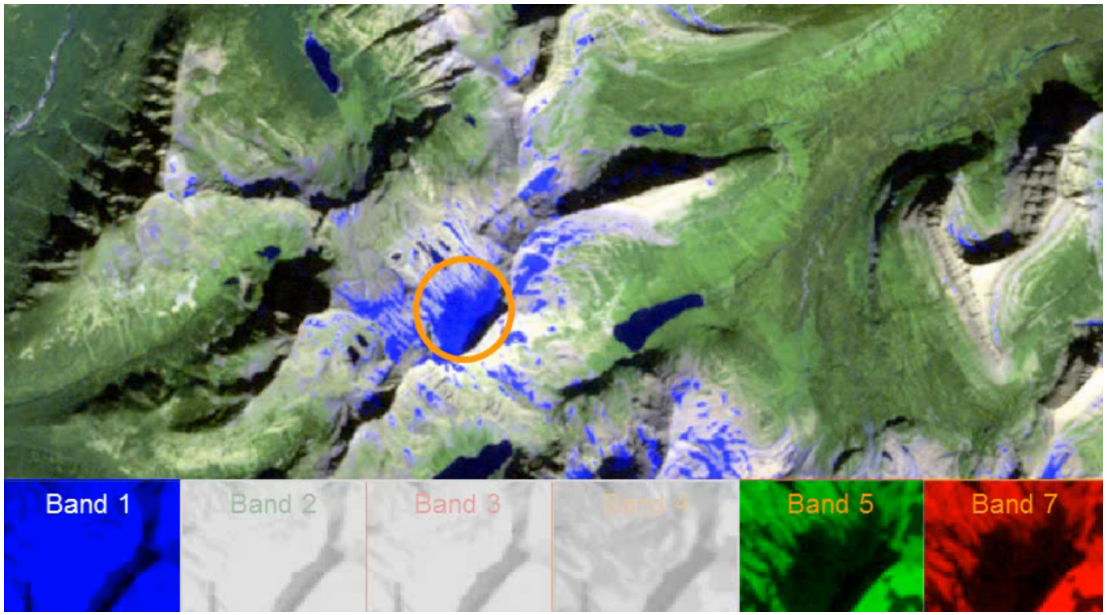
Satellite image of Sperry Glacier (Montana)



- Snow is only "white" in bands 1-4 of the spectrum (where snow strongly reflects energy)
- Snow absorbs radiation in other parts of the spectrum

Remote sensing

Band combinations – snow



source: `https://www.youtube.com/watch?v=zesEJouvNU4`

- We can use knowledge of how different land surfaces reflect energy from different portions of the spectrum
- We can assign colors to different bands and combine bands to build composite images
- Assigning the three primary colors to bands 1, 5, and 7, we get a composite image that lets us very clearly "see" patches of snow

Remote sensing

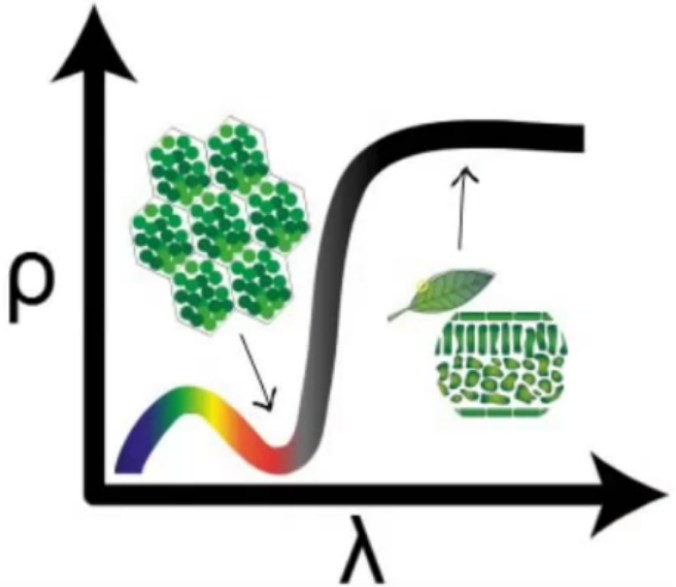
Band combinations – properties of different bands

- **Red band**
  - Absorbed by chlorophyll – little reflectance from healthy vegetation
  - Also absorbed by water – water bodies within this band appear fairly dark
- **Green band**
  - Better reflected by water
  - Can use to distinguish clear and turbid water
  - Strongly reflected by vegetation
- **Blue band**
  - Penetrates clear water well, used for shallow bathymetric mapping
  - Can use to distinguish soil and vegetation and deciduous from coniferous vegetation
- Red + Green + Blue band = "true color" composite
- true color composites tend to be low contrast and somewhat hazy looking due to blue band's susceptibility to scatter by the atmosphere
- **Near Infrared band**
  - Strongly absorbed by healthy vegetation

Remote sensing

NDVI

The material in this section is taken from https://www.youtube.com/watch?v=rxOMhQwApMc and https://en.wikipedia.org/wiki/Normalized_difference_vegetation_index



λ = wavelenth, ρ = reflectance
source: https://www.youtube.com/watch?v=rxOMhQwApMc

- The Normalized Difference Vegetation Index is commonly used in remote sensing to identify vegegation and measure its health and vitality
  - Presence of chlorophyl causes light absorption in the red part of the spectrum (plants use radiation from red portion of spectrum as energy for photosynthesis)
  - Plant leaf cells re-emit solar radiation in near-infrared portion of the spectrum because energy at wavelengths longer than $\approx$ 700nm is too small to synthesize organic molecules. Absorbing these wavelengths too strongly would result in the plant overheating and tissue damage
  - The result of these is a particular spectral signature of vegetation
  - When vegetation is less healthy, more sparse, more red and less infrared light gets reflected

Remote sensing

NDVI

$$\text{NDVI} = \frac{\text{NIR - RED}}{\text{NIR} + \text{RED}}$$

- normalize by sum of NIR and RED, hence $|\text{NDVI}| \in [0, 1]$
- healthy vegetation, NIR radiation is reflected (large DN) and RED is absorbed (small DN), hence NDVI ≈ 1
- less healthy vegetation, NIR smaller, RED larger DN, NDVI smaller, but usually still positive
- NDVI can be used to identify other features, e.g. water, which absorbs almost all NIR light and some but not all RED light. Hence $NDVI < 0$.

Google Earth Engine

Introduction

Google Earth Engine (GEE) is a cloud computing platform combining

1) Vast collection of satellite images and other geospatial data, together with great documentation:

   `https://developers.google.com/earth-engine/datasets/`

- You can also upload your own data, as we will see shortly

2) Convenient coding environment that runs within the browser and allows you to visualize data as well as analyse and transform them with a broad range of algorithms

3) Ability to run analysis on google servers $\rightarrow$ analysing huge datasets becomes possible (though there are still query limits)

Google Earth Engine

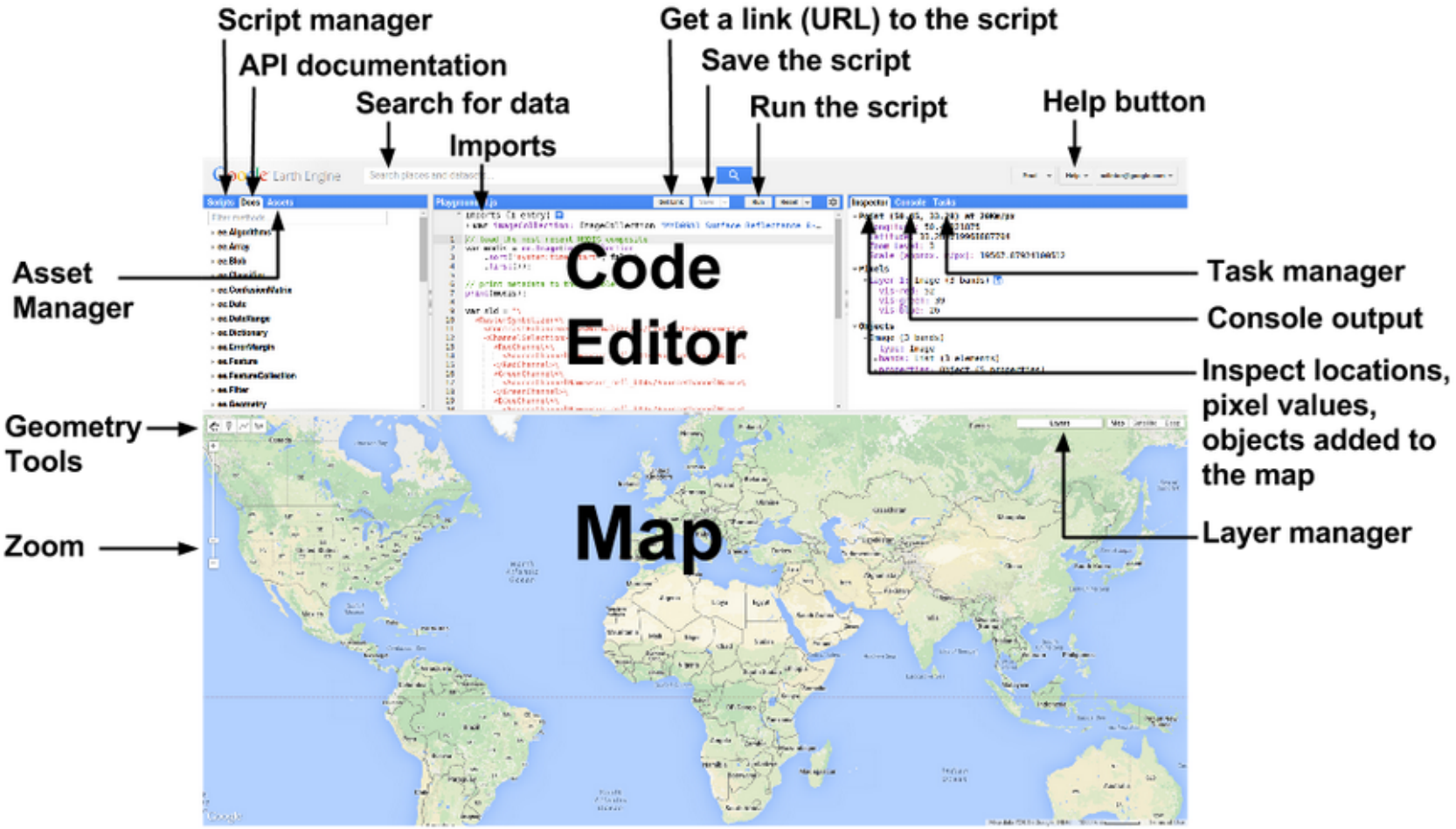
Sign-up

GEE is free to use for non-commercial purposes

- create a google account if you don't already have one
- register for google earth engine: `https://signup.earthengine.google.com/`
- go to `code.earthengine.google.com` and log in
- if needed, grant the relevant permissions

Google Earth Engine

Interface



- source: GEARS lab github page: `https://github.com/geospatialeco/GEARS`
- see also the GEARS lab page `https://www.gears-lab.com/`
  and youtube channel: `https://www.youtube.com/channel/UCPZMj2ykE9pgJGV1r0kXQMg`
- both maintained by Shaun R Levick, Assoc. Prof of Remote Sensing at Charles Darwin University, RIEL
- If you want to get deeper into GEE, this (in addition to the excellent official documentation) is a great starting point

Google Earth Engine

Finding and loading data

- Go to the search bar at the top (next to "Google Earth Engine") and type "nighttime lights"
- select *DMSP OLS: Nighttime Lights Time Series Version 4, Defense Meteorological Program Operational Linescan System* (don't click "Import" yet)
- note resolution, information on bands, terms of use
- click 

- nothing seems to happen (but no error either...) except new entry in the script window
- click on `imageCollection` and rename to `lights`
- in line 1 of the script window, type (don't forget the semi-colon, this is Javascript):

```
print(lights, 'nightlights image collection');
```

- click 

- In the "Console"-tab on the left, a new `ImageCollection` appears
- This contains information on the loaded images, organized in JSON format (Javascript Object Notation)

Google Earth Engine

Displaying data

- We have imported and displayed information on all of the DMSP OLS data, organized into an image collection (think of this literally as a stack of images)
- Let's try to visualize a particular year
- type

```
var lights00 = ee.Image(lights.filter(ee.Filter.date('2000'))
                              .filterMetadata('system:index', 'contains', 'F15')
                              .select('stable_lights')
                              .first());
```

- then type

```
Map.addLayer(lights00);
```

- This adds the data to the map. If you click on 

, you can see which layers are loaded, turn loaded layers on and off, adjust their opacity and customize their appearance (⚙)
- We can also change visualization in the script. Type
- `var visParams = {min: 0, max: 63, palette: ['blue', 'yellow']};`
- and replace the line `Map.addLayer(lights00);` with
- `Map.addLayer(lights00, visParams, 'night lights in 2000');`
- Click on the "Inspector" tab in the right pane, and click anywhere on the map (see that the cursor transforms into a crosshair). Information on the points you click on appear. If you have more than one layer loaded, you will get information on all layers at that point.

Calculating NDVI for admin-2 regions in Ghana using Landsat-8 data

- Say we want to investigate deforestation in Ghana
- get the admin-2 boundaries for Ghana from google drive
- Alternatively, go here: `https://gadm.org/download_country_v3.html`
- Download the Shapefile for Ghana
- unzip and delete everything except the files starting with *gadm36_GHA_2*
- back in GEE, in the left pane, click on the "Assets" tab
- "New" → "Table upload" → Shapefiles → "Select" → select all the admin-2 files (not just the shapefile!)
- choose an asset-ID (if you choose a different one from the default *gadm36_GHA_2*, you will need to adjust the script we will be working on) and click OK.
- Open the "Tasks" tab in the right pane, the asset is being ingested. This takes several minutes.
- In the script window, click on the downward-facing arrow next to "Reset" and click "Clear script" (you can save what you have done before).
- Copy and paste the code in *ghana_admin2_ndvi.js* that we have provided for you. We will go through it together.

Google Earth Engine

Land cover classification

Classification

- Start from raw satellite data
- Create training data of labelled classes, e.g. ['Water', 'Rock', 'Forest', 'Grassland'...] (subset of the raw data)
- Train a classifier
- Label the full dataset

Accuracy assessment

- Before running the classifier, also create a separate validation dataset with known labels
- Check prediction vs. actual labels after training the classifier
- Compute accuracy $\left(\frac{N^{\text{true pos}}+N^{\text{true neg}}}{N^{\text{all}}}\right)$ and confusion matrix

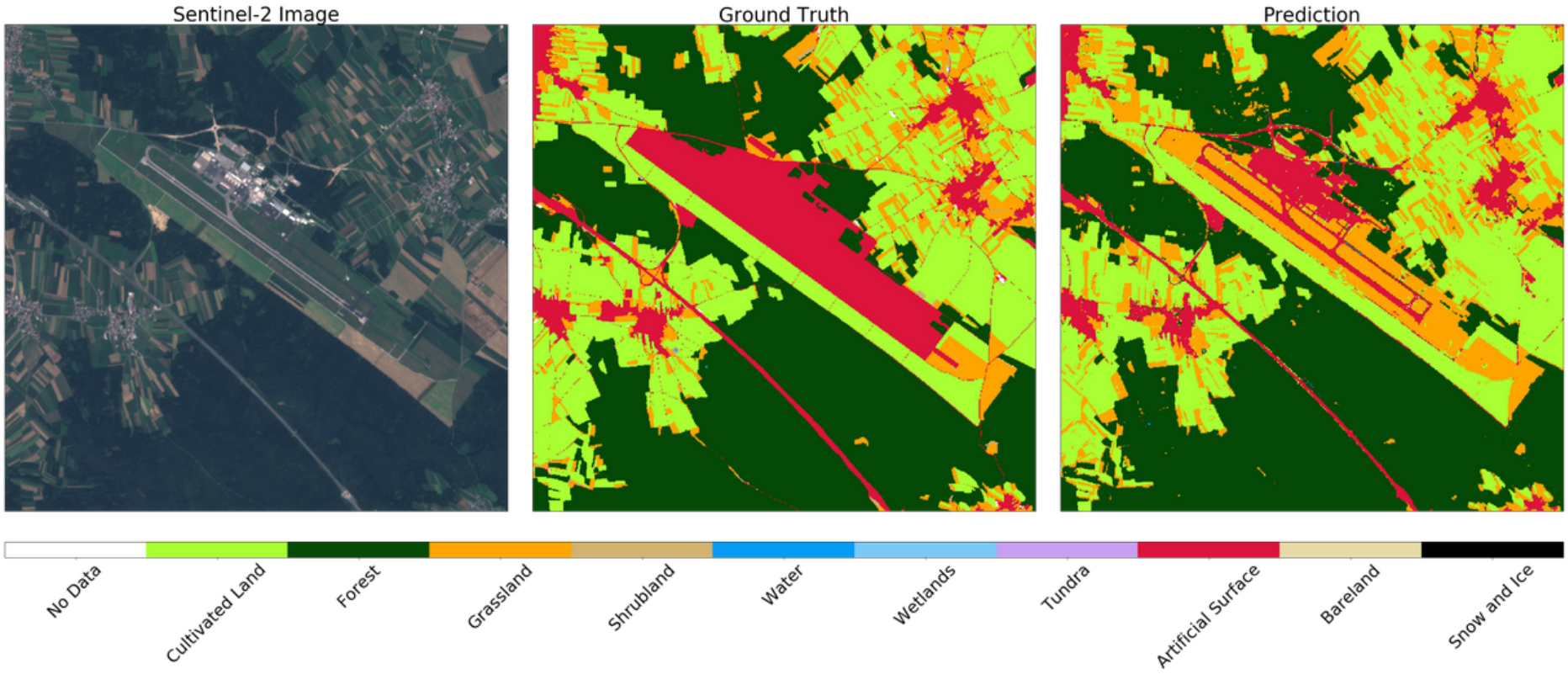
For the hair-splitters...

- land use (human uses of land) and land cover (physical environment) are not strictly the same thing
- see here if you are interested: `http://www.fao.org/3/x0596e/x0596e01e.htm`

Google Earth Engine

Land cover classification

Land cover classification with Sentinel 2 data



source: https://medium.com/sentinel-hub/land-cover-classification-with-eo-learn-part-2-bd9aa86f8500

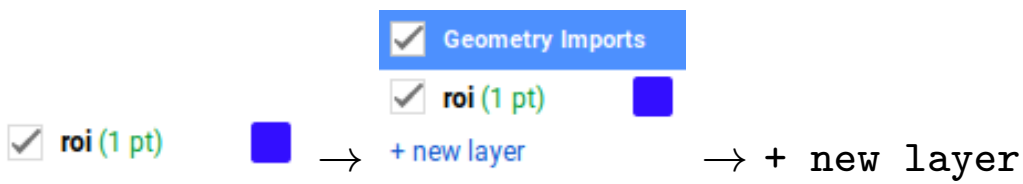
Google Earth Engine

Land cover classification

- Clear your script and copy and paste the code from *landcover_classification.js* into the script window
- The code is by Shaun R. Levick of GEARS Lab, see here for the source: `https://github.com/geospatialeco/GEARS/blob/master/Intro_RS_Lab7.md`
- Before running the script, we need to do a couple of things by hand

Adding geometries for region of interest, training and validation sets

- Go into the search bar at the top and search for "London", select "London, UK" and click on it to move the map to London
- In the map pane, on the top-left, click on the teardrop icon and place one in the center of London
- Rename the imported variable to `roi` (for region of interest)
- Next, change to the rectangle icon and hover next to it over



- Do this 7 more times so you have added 8 geometries plus 1 point feature (the roi) in total.

Google Earth Engine

Land cover classification

Modifying the geometries for the training and validation data

- In the script window, rename the first geometry under `roi` to `tW` (short for "test water")
- Similarly, rename the next seven geometries so you have test and validation features for water, urban, forest, and agriculture
- Click on  next to `tW` and change `Import as` to `FeatureCollection`
- Click on `+ Add property` → fill in *lc* into `name` and *0* for the value. This is the value of the landcover feature for the "water" class
- Do the same for the other seven imported geometries (water: 0, urban: 1, forest: 2, agriculture: 3; you can choose different values, just make sure training and validation sets get the same numbers for the same class)
- Run the code up to `Map.centerObject(roi, 8);` (comment out the rest) – we will discuss first what this is doing

Drawing polygons for training and validation sets

- Back in the map pane, pan to an area with water (south coast of England), select again the  , make sure the `tW` feature is active, and draw a small rectangle into the ocean
- How small? You will see – if your training data is too large, GEE will complain
- Do the same thing for the remaining training and validation sets.
- Use the loaded landsat image and the google's own satellite data as guidance for where to draw

Running the classifier

- Run the code up to `Map.addLayer(classified, visParams, 'classification');`
- How good is the classification? Where are the problems?
- This uses a classification and regression tree (CART) to do classification.
- Read the documentation for different classifiers (play with parameters), try different ones.
- What can we try to do to improve?

Assessing accuracy

- Run the rest of the code
- This produces a simple accuracy metric (may be very high, but perhaps not very informative...)
- Better to look at the confusion matrix