Day 2: Earth Observation and Generative AI – Complete Instructor Guide

Overview & Timing Adjustments

- **09:00–09:30**: Recap + Earth Observation for Health
- **09:30–10:30**: Lab 3 - NDVI & Rainfall Visualization
- **10:30–11:00**: Break
- **11:00–12:00**: Lab 4 - ChatGPT for GEE (15 min lecture + 45 min hands-on)
- **12:00–13:00**: Lunch
- **13:00–14:30**: Lab 5 - K-means Clustering (includes 10 min coffee stretch)
- **14:30–15:30**: Export and Visualize in QGIS
- **15:30–16:00**: Reflection and Discussion

Lab 3: Visualizing NDVI and Rainfall in GEE (09:30-10:30)

Pre-Lab Setup

```javascript
// Instructor should have this ready to share
// Base setup code for all participants
var uganda = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_3')
  .filter(ee.Filter.eq('NAME_0', 'Uganda'));
var ugandaBounds = uganda.geometry().bounds();
Map.centerObject(uganda, 7);
```

Exercise 1: Basic NDVI Visualization (15 min)

Step 1: Load and filter MODIS NDVI

javascript

```javascript
// Load MODIS NDVI data
var modis = ee.ImageCollection('MODIS/006/MOD13Q1')
  .filterDate('2024-01-01', '2024-12-31')
  .filterBounds(uganda)
  .select('NDVI');

// Scale NDVI values (MODIS uses scale factor of 0.0001)
var ndviScaled = modis.map(function(img) {
  return img.multiply(0.0001)
    .copyProperties(img, ['system:time_start']);
});

// Get median NDVI for the year
var ndviMedian = ndviScaled.median().clip(uganda);
```

Step 2: Apply preset palettes (Enhancement)

javascript

```javascript
// Preset palette options
var palettes = {
  vegetation: ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163',
               '99B718', '74A901', '66A000', '529400', '3E8601',
               '207401', '056201', '004C00', '023B01', '012E01', '011D01'],
  risk: ['2166AC', '4393C3', '92C5DE', 'D1E5F0', 'F7F7F7',
         'FDDBC7', 'F4A582', 'D6604D', 'B2182B'],
  simple: ['brown', 'yellow', 'green']
};

// Create UI selector for palettes
var paletteSelect = ui.Select({
  items: Object.keys(palettes),
  value: 'vegetation',
  onChange: function(key) {
    Map.layers().get(0).setVisParams({
      min: 0, max: 1, palette: palettes[key]
    });
  }
});

// Add to map
Map.add(paletteSelect);
Map.addLayer(ndviMedian, {min: 0, max: 1, palette: palettes.vegetation}, 'NDVI');
```

Exercise 2: Rainfall Integration (15 min)

Step 3: Load CHIRPS rainfall data

javascript

```javascript
// Load CHIRPS rainfall
var chirps = ee.ImageCollection('UCSB-CHG/CHIRPS/DAILY')
  .filterDate('2024-01-01', '2024-12-31')
  .filterBounds(uganda);

// Calculate total annual rainfall
var annualRainfall = chirps.sum().clip(uganda);

// Create split panel map
var leftMap = ui.Map();
var rightMap = ui.Map();

// Configure maps
leftMap.addLayer(ndviMedian, {min: 0, max: 1, palette: palettes.vegetation}, 'NDVI');
rightMap.addLayer(annualRainfall, {min: 0, max: 3000, palette: palettes.risk}, 'Rainfall');

// Link the maps
var linker = ui.Map.Linker([leftMap, rightMap]);
leftMap.centerObject(uganda, 7);

// Create split panel
var splitPanel = ui.SplitPanel({
  firstPanel: leftMap,
  secondPanel: rightMap,
  orientation: 'horizontal',
  wipe: true
});

// Replace the root with split panel
ui.root.clear();
ui.root.add(splitPanel);
```

Exercise 3: District Summary Export (10 min)

Step 4: Calculate district statistics

javascript

```javascript
// Load district boundaries
var districts = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_2')
  .filter(ee.Filter.eq('NAME_0', 'Uganda'));

// Function to extract mean values per district
var extractStats = function(feature) {
  var ndviMean = ndviMedian.reduceRegion({
    reducer: ee.Reducer.mean(),
    geometry: feature.geometry(),
    scale: 250,
    maxPixels: 1e9
  });

  var rainfallMean = annualRainfall.reduceRegion({
    reducer: ee.Reducer.mean(),
    geometry: feature.geometry(),
    scale: 5000,
    maxPixels: 1e9
  });

  return feature.set({
    'ndvi_mean': ndviMean.get('NDVI'),
    'rainfall_mean': rainfallMean.get('precipitation'),
    'district': feature.get('NAME_2')
  });
};

// Apply to all districts
var districtStats = districts.map(extractStats);

// Export to Drive
Export.table.toDrive({
  collection: districtStats,
  description: 'Uganda_District_Environmental_Stats',
  fileFormat: 'CSV',
  selectors: ['district', 'ndvi_mean', 'rainfall_mean']
});
```

Exercise 4: QA Visualization (10 min)

Step 5: Create validation charts

```javascript
// Select two contrasting points
var wetPoint = ee.Geometry.Point([32.5, 0.5]); // Near Lake Victoria
var dryPoint = ee.Geometry.Point([34.0, 3.5]); // Karamoja region

// Time series chart
var chart = ui.Chart.image.series({
  imageCollection: ndviScaled.select('NDVI'),
  region: wetPoint,
  reducer: ee.Reducer.mean(),
  scale: 250
}).setOptions({
  title: 'NDVI Time Series - Wet vs Dry Regions',
  vAxis: {title: 'NDVI'},
  hAxis: {title: 'Date'},
  series: {
    0: {color: 'green', label: 'Wet Region'},
    1: {color: 'brown', label: 'Dry Region'}
  }
});

print(chart);
```

Why These Variables? (5 min debrief)

Instructor talking points:

- NDVI indicates vegetation density - higher values suggest more mosquito breeding habitat
- Rainfall creates standing water - critical for mosquito larvae
- Combined, they predict malaria transmission potential
- Temporal patterns matter - lag between rainfall and cases

Lab 4: Using ChatGPT to Prompt and Write GEE Code (11:00-12:00)

Mini-Lecture: Prompt Patterns (15 min)

Distribute this cheat sheet:

markdown

```
# ChatGPT Prompt Patterns for GEE

## 1. Diagnose Pattern
"I'm getting this error in Google Earth Engine: [paste error].
My code is: [paste code]. What's wrong and how do I fix it?"

## 2. Refactor Pattern
"Optimize this GEE code for better performance: [paste code].
Focus on reducing computation time and memory usage."

## 3. Explain-as-Analogy Pattern
"Explain how ee.Reducer.frequencyHistogram() works using
a real-world analogy that health workers would understand."

## 4. Translate Pseudocode Pattern
"Convert this logic to GEE JavaScript:
- Load rainfall data for Uganda
- For each month, find areas with >100mm rain
- Count how many months each pixel exceeds threshold"

## 5. Rate-Limit Aware Pattern
"Rewrite this script to batch-export by district to avoid
GEE computation timeouts: [paste code]"
```

Hands-On Exercises (45 min)

Exercise 1: Debug a Broken Script (15 min)

Give participants this broken code:

javascript

```javascript
// BROKEN CODE - FIX WITH CHATGPT
var uganda = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_3')
  .filter(ee.Filter.eq('NAME_0', 'Uganda');  // Missing parenthesis

var ndvi = ee.ImageCollection('MODIS/006/MOD13Q1')
  .filterdate('2024-01-01', '2024-12-31')  // Wrong case
  .filterBounds(uganda)
  .select(['NDVI']);

var mean_ndvi = ndvi.mean();  // Forgot to scale

Map.addLayer(mean_ndvi, {}, 'NDVI');  // No visualization parameters

// Try to export without clipping
Export.image.toDrive({
  image: mean_ndvi,
  description: 'Uganda_NDVI',
  scale: 250
});
```

Expected ChatGPT fixes:

javascript

```javascript
// FIXED CODE
var uganda = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_3')
  .filter(ee.Filter.eq('NAME_0', 'Uganda')); // Fixed parenthesis

var ndvi = ee.ImageCollection('MODIS/006/MOD13Q1')
  .filterDate('2024-01-01', '2024-12-31')  // Fixed case
  .filterBounds(uganda)
  .select(['NDVI']);

// Scale NDVI values
var ndviScaled = ndvi.map(function(image) {
  return image.multiply(0.0001);
});

var mean_ndvi = ndviScaled.mean().clip(uganda); // Added clip

Map.addLayer(mean_ndvi, {min: 0, max: 1, palette: ['white', 'green']}, 'NDVI');

// Export with region specified
Export.image.toDrive({
  image: mean_ndvi,
  description: 'Uganda_NDVI',
  scale: 250,
  region: uganda.geometry().bounds(),
  maxPixels: 1e13
});
```

Exercise 2: Build Mosquito Breeding Site Detector (15 min)

Prompt sequence for participants:

1. **First prompt**: "Write a GEE script that identifies potential mosquito breeding sites by finding areas with NDVI > 0.6 and rainfall > 50mm in the last month"
2. **Second prompt**: "Add error handling for missing data and cloud masking to this script: [paste result]"
3. **Third prompt**: "Optimize this for large areas by using reduceRegions instead of pixel-wise operations"

Expected final code:

```
javascript
```

```
// Function to identify breeding sites
function identifyBreedingSites(startDate, endDate) {
  // Load data
  var uganda = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_3')
    .filter(ee.Filter.eq('NAME_0', 'Uganda'));

  // Get NDVI
  var ndvi = ee.ImageCollection('MODIS/006/MOD13Q1')
    .filterDate(startDate, endDate)
    .filterBounds(uganda)
    .select('NDVI')
    .map(function(img) {
      return img.multiply(0.0001)
        .updateMask(img.select('NDVI').gt(0)); // Remove invalid pixels
    })
    .median();

  // Get rainfall
  var rainfall = ee.ImageCollection('UCSB-CHG/CHIRPS/DAILY')
    .filterDate(startDate, endDate)
    .filterBounds(uganda)
    .sum();

  // Identify high-risk areas
  var highNDVI = ndvi.gt(0.6);
  var highRainfall = rainfall.gt(50);
  var breedingSites = highNDVI.and(highRainfall);

  // Calculate area by district
  var districts = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_2')
    .filter(ee.Filter.eq('NAME_0', 'Uganda'));

  var areaByDistrict = breedingSites.multiply(ee.Image.pixelArea())
    .reduceRegions({
      collection: districts,
      reducer: ee.Reducer.sum(),
      scale: 250
    });

  return {
    riskMap: breedingSites.selfMask().clip(uganda),
    statistics: areaByDistrict
  };
```

```
}

// Run for last month
var endDate = ee.Date(Date.now());
var startDate = endDate.advance(-1, 'month');
var results = identifyBreedingSites(startDate, endDate);

// Visualize
Map.addLayer(results.riskMap, {palette: ['red']}, 'Breeding Sites');
print('Area by district:', results.statistics);
```

Exercise 3: Create Reusable Functions (15 min)

Prompt: "Create a reusable function library for common GEE operations in health mapping: getMedianNDVI, getTotalRainfall, and standardizeImage"

Expected result:

```
javascript
```

```javascript
// Reusable function library for health mapping
var healthMapUtils = {
  // Get median NDVI for a region and time period
  getMedianNDVI: function(region, startDate, endDate) {
    return ee.ImageCollection('MODIS/006/MOD13Q1')
      .filterDate(startDate, endDate)
      .filterBounds(region)
      .select('NDVI')
      .map(function(img) {
        return img.multiply(0.0001)
          .copyProperties(img, ['system:time_start']);
      })
      .median()
      .clip(region);
  },

  // Get total rainfall
  getTotalRainfall: function(region, startDate, endDate) {
    return ee.ImageCollection('UCSB-CHG/CHIRPS/DAILY')
      .filterDate(startDate, endDate)
      .filterBounds(region)
      .sum()
      .clip(region);
  },

  // Standardize image to 0-1 range
  standardizeImage: function(image, bandName) {
    var band = image.select(bandName);
    var minMax = band.reduceRegion({
      reducer: ee.Reducer.minMax(),
      geometry: image.geometry(),
      scale: 1000,
      maxPixels: 1e9
    });

    var min = ee.Number(minMax.get(bandName + '_min'));
    var max = ee.Number(minMax.get(bandName + '_max'));

    return band.subtract(min).divide(max.subtract(min))
      .rename(bandName + '_normalized');
  }
};
```

```
// Example usage
var uganda = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_3')
  .filter(ee.Filter.eq('NAME_0', 'Uganda'));

var ndvi = healthMapUtils.getMedianNDVI(uganda, '2024-01-01', '2024-03-31');
var rainfall = healthMapUtils.getTotalRainfall(uganda, '2024-01-01', '2024-03-31');
var ndviNorm = healthMapUtils.standardizeImage(ndvi, 'NDVI');

Map.addLayer(ndviNorm, {min: 0, max: 1, palette: ['brown', 'yellow', 'green']}, 'NDVI Normalize
```

Lab 5: AI-Based Clustering for Malaria Risk Mapping (13:00-14:30)

Exercise 1: Prepare Multi-band Stack with Scaling (20 min)

Step 1: Create properly scaled multi-band image

```
javascript
```

```
// Load all required data
var uganda = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_3')
  .filter(ee.Filter.eq('NAME_0', 'Uganda'));

// Environmental variables
var ndvi = healthMapUtils.getMedianNDVI(uganda, '2024-01-01', '2024-03-31');
var rainfall = healthMapUtils.getTotalRainfall(uganda, '2024-01-01', '2024-03-31');

// Add elevation
var elevation = ee.Image('USGS/SRTMGL1_003').select('elevation').clip(uganda);

// Add temperature (optional)
var temperature = ee.ImageCollection('MODIS/006/MOD11A1')
  .filterDate('2024-01-01', '2024-03-31')
  .select('LST_Day_1km')
  .map(function(img) {
    return img.multiply(0.02).subtract(273.15); // Convert to Celsius
  })
  .mean()
  .clip(uganda);

// ChatGPT-assisted min-max scaling
var scaleImage = function(image, bandName) {
  var band = image.select(bandName);
  var minMax = band.reduceRegion({
    reducer: ee.Reducer.minMax(),
    geometry: uganda.geometry(),
    scale: 1000,
    maxPixels: 1e9,
    bestEffort: true
  });

  var min = ee.Number(minMax.get(bandName + '_min'));
  var max = ee.Number(minMax.get(bandName + '_max'));

  return band.subtract(min).divide(max.subtract(min))
    .rename(bandName + '_scaled');
};

// Scale all bands
var ndviScaled = scaleImage(ndvi, 'NDVI');
var rainfallScaled = scaleImage(rainfall, 'precipitation');
var elevationScaled = scaleImage(elevation, 'elevation');
```

```
var tempScaled = scaleImage(temperature, 'LST_Day_1km');

// Create multi-band image
var composite = ndviScaled
  .addBands(rainfallScaled)
  .addBands(elevationScaled)
  .addBands(tempScaled);

print('Composite bands:', composite.bandNames());
```

Coffee Break (10 min)

Exercise 2: K-means Clustering (30 min)

Step 2: Implement clustering with validation

javascript

```javascript
// Prepare training data
var training = composite.sample({
  region: uganda,
  scale: 1000,
  numPixels: 5000
});

// Instantiate clusterer and train
var clusterer = ee.Clusterer.wekaKMeans({
  nClusters: 5,
  seed: 42  // For reproducibility
}).train(training);

// Apply clusterer
var clustered = composite.cluster(clusterer);

// Assign risk levels based on cluster characteristics
var clusterMeans = ee.FeatureCollection(ee.List.sequence(0, 4).map(function(i) {
  var cluster = clustered.updateMask(clustered.eq(i));
  var means = composite.updateMask(cluster).reduceRegion({
    reducer: ee.Reducer.mean(),
    geometry: uganda.geometry(),
    scale: 5000,
    maxPixels: 1e9
  });

  return ee.Feature(null, means).set('cluster', i);
}));

print('Cluster characteristics:', clusterMeans);

// Remap clusters to risk levels (0=lowest, 4=highest)
// This should be adjusted based on domain knowledge
var riskLevels = clustered.remap([0, 1, 2, 3, 4], [2, 4, 1, 3, 0]);

// Visualize with health-appropriate colors
var riskPalette = ['2166AC', '67A9CF', 'F7F7F7', 'FDDBC7', 'B2182B'];
Map.addLayer(riskLevels, {min: 0, max: 4, palette: riskPalette}, 'Malaria Risk Levels');
```

Exercise 3: Spatial Cross-Validation (20 min)

Step 3: Hold-out validation

javascript

```javascript
// Define hold-out region (e.g., Northern Uganda)
var northernRegion = ee.FeatureCollection('UCSB-CHG/GADM/gadm36_1')
  .filter(ee.Filter.eq('NAME_0', 'Uganda'))
  .filter(ee.Filter.eq('NAME_1', 'Northern'));

var trainingRegion = uganda.geometry().difference(northernRegion.geometry());

// Retrain on subset
var trainingSubset = composite.sample({
  region: trainingRegion,
  scale: 1000,
  numPixels: 5000
});

var clustererSubset = ee.Clusterer.wekaKMeans({
  nClusters: 5,
  seed: 42
}).train(trainingSubset);

// Apply to full area
var clusteredValidation = composite.cluster(clustererSubset);

// Compare clusters in hold-out region
var comparisonMap = clustered.addBands(clusteredValidation)
  .select(['cluster', 'cluster_1'], ['original', 'validated']);

Map.addLayer(comparisonMap.select('original').clip(northernRegion),
  {min: 0, max: 4, palette: ['red', 'orange', 'yellow', 'lightgreen', 'darkgreen']},
  'Original Clusters - North');

Map.addLayer(comparisonMap.select('validated').clip(northernRegion),
  {min: 0, max: 4, palette: ['red', 'orange', 'yellow', 'lightgreen', 'darkgreen']},
  'Validated Clusters - North');
```

Exercise 4: Validate Against "Ground Truth" (20 min)

Step 4: Load and compare with malaria prevalence data

```javascript
// Simulated malaria prevalence data (in practice, use real data)
// For workshop, create synthetic data based on environmental conditions
var createSyntheticPrevalence = function() {
  // Higher risk where NDVI is moderate and rainfall is high
  var prevalence = ndvi.multiply(2)
    .add(rainfall.divide(1000))
    .add(elevation.divide(-2000))  // Lower elevation = higher risk
    .add(temperature.multiply(-0.05))  // Optimal temp around 25°C
    .unitScale(0, 3)  // Scale to 0-1
    .rename('prevalence');

  return prevalence;
};

var malariaPrevalence = createSyntheticPrevalence();

// Sample prevalence at cluster centroids
var validation = clustered.addBands(malariaPrevalence)
  .stratifiedSample({
    numPoints: 100,
    classBand: 'cluster',
    region: uganda,
    scale: 1000
  });

// Create scatter plot
var chart = ui.Chart.feature.byFeature(validation, 'cluster', 'prevalence')
  .setChartType('ColumnChart')
  .setOptions({
    title: 'Mean Malaria Prevalence by Risk Cluster',
    hAxis: {title: 'Cluster ID'},
    vAxis: {title: 'Prevalence'},
    colors: riskPalette
  });

print(chart);
```

Wrap-up Sheet (10 min)

Teams complete this summary:

```markdown
# Cluster Analysis Summary

Team Members: _____

## Cluster Characteristics
| Cluster | NDVI | Rainfall | Elevation | Temperature | Assigned Risk |
|---------|------|----------|-----------|-------------|---------------|
| 0       |      |          |           |             |               |
| 1       |      |          |           |             |               |
| 2       |      |          |           |             |               |
| 3       |      |          |           |             |               |
| 4       |      |          |           |             |               |

## Interpretation
Highest risk areas are characterized by: _____
Lowest risk areas are characterized by: _____

## Validation Results
Correlation with prevalence data: _____
Spatial stability (North vs. South): _____
```

Export Bridge: Storyboard Documentation (14:30-15:30)

Auto-generate Documentation with ChatGPT

Prompt for participants: "Generate a markdown header for my GEE script that documents:

1. Data sources with citations
2. Processing steps
3. Assumptions and limitations
4. Export parameters"

Expected output:

```javascript
/**
 * MALARIA RISK MAPPING - UGANDA
 * ==============================
 *
 * DATA SOURCES:
 * - MODIS NDVI (MOD13Q1): Didan, K. (2015). MOD13Q1 MODIS/Terra Vegetation Indices
 *   16-Day L3 Global 250m SIN Grid V006. NASA EOSDIS Land Processes DAAC.
 * - CHIRPS Rainfall: Funk, et al. (2015). The climate hazards infrared precipitation
 *   with stations—a new environmental record for monitoring extremes. Scientific Data.
 * - SRTM Elevation: Farr, T.G., et al. (2007). The Shuttle Radar Topography Mission.
 *   Rev. Geophys., 45, RG2004.
 *
 * PROCESSING CHAIN:
 * 1. Load environmental variables (NDVI, rainfall, elevation, temperature)
 * 2. Apply min-max scaling to standardize inputs
 * 3. Create multi-band composite
 * 4. Apply k-means clustering (k=5)
 * 5. Assign risk levels based on cluster characteristics
 * 6. Validate against synthetic prevalence data
 *
 * ASSUMPTIONS:
 * - Environmental conditions are primary drivers of malaria risk
 * - 3-month average represents typical seasonal conditions
 * - Linear scaling preserves relative relationships
 *
 * LIMITATIONS:
 * - Does not account for human factors (population density, interventions)
 * - Cluster assignments may vary with different initial conditions
 * - Validation uses synthetic data for workshop purposes
 *
 * EXPORT:
 * - Format: GeoTIFF
 * - Resolution: 250m
 * - Projection: EPSG:4326
 * - Bands: risk_level (0-4, where 4 is highest risk)
 */

// Main script starts here...
```

Final Export Code

javascript

```javascript
// Export risk map
Export.image.toDrive({
  image: riskLevels.byte(),  // Convert to byte for smaller file size
  description: 'Uganda_Malaria_Risk_Map_' + ee.Date(Date.now()).format('YYYY_MM_dd').getInfo(),
  folder: 'GEE_Workshop',
  region: uganda.geometry().bounds(),
  scale: 250,
  crs: 'EPSG:4326',
  maxPixels: 1e13
});

// Export cluster statistics
Export.table.toDrive({
  collection: clusterMeans,
  description: 'Cluster_Statistics',
  fileFormat: 'CSV'
});

// Export validation points
Export.table.toDrive({
  collection: validation,
  description: 'Validation_Points',
  fileFormat: 'CSV',
  selectors: ['cluster', 'prevalence', '.geo']
});
```

Optional Stretch Tasks

1. Auto-generate Interactive Leaflet Map

ChatGPT Prompt: "Convert this GEE export URL to a Leaflet web map with popup showing risk level"

Expected HTML:

```
html
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Uganda Malaria Risk Map</title>
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />
  <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
  <script src="https://unpkg.com/georaster"></script>
  <script src="https://unpkg.com/georaster-layer-for-leaflet"></script>
  <style>
    #map { height: 600px; }
    .legend {
      background: white;
      padding: 10px;
      border-radius: 5px;
      box-shadow: 0 1px 5px rgba(0,0,0,0.4);
    }
  </style>
</head>
<body>
  <div id="map"></div>
  <script>
    // Initialize map
    var map = L.map('map').setView([1.373333, 32.290275], 7);

    // Add base layer
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
      attribution: '© OpenStreetMap contributors'
    }).addTo(map);

    // Load GeoTIFF (participants add their export URL)
    fetch('YOUR_GEOTIFF_URL')
      .then(response => response.arrayBuffer())
      .then(arrayBuffer => {
        parseGeoraster(arrayBuffer).then(georaster => {
          const layer = new GeoRasterLayer({
            georaster: georaster,
            opacity: 0.7,
            pixelValuesToColorFn: values => {
              const riskColors = ['#2166AC', '#67A9CF', '#F7F7F7', '#FDDBC7', '#B2182B'];
              const value = values[0];
              return value >= 0 && value <= 4 ? riskColors[value] : null;
            },
            resolution: 256
```

```
          });
          layer.addTo(map);
          map.fitBounds(layer.getBounds());
        });
      });

    // Add Legend
    var legend = L.control({position: 'bottomright'});
    legend.onAdd = function(map) {
      var div = L.DomUtil.create('div', 'legend');
      div.innerHTML = '<h4>Malaria Risk</h4>' +
        '<i style="background: #B2182B"></i> Very High<br>' +
        '<i style="background: #FDDBC7"></i> High<br>' +
        '<i style="background: #F7F7F7"></i> Moderate<br>' +
        '<i style="background: #67A9CF"></i> Low<br>' +
        '<i style="background: #2166AC"></i> Very Low';
      return div;
    };
    legend.addTo(map);
  </script>
</body>
</html>
```

Instructor Notes

Common Issues and Solutions

1. **GEE Computation Timeout**
   - Solution: Reduce scale parameter, use bestEffort: true, or batch by region
2. **ChatGPT Rate Limits**
   - Solution: Have participants work in pairs, prepare backup responses
3. **Visualization Not Showing**
   - Solution: Check geometry bounds, ensure clipping is applied
4. **Export Taking Too Long**
   - Solution: Reduce resolution or export smaller regions

Assessment Rubric

Component	Excellent (3)	Good (2)	Needs Improvement (1)
NDVI/Rainfall Map	Both layers properly scaled and visualized	One layer has issues	Multiple errors
ChatGPT Usage	Effective prompting, meaningful iterations	Basic prompting	Struggled with prompts
Clustering	All 5 clusters with clear interpretation	Clusters created but unclear	Failed to complete
Documentation	Complete storyboard with citations	Partial documentation	Minimal documentation

Additional Resources

- GEE Datasets Catalog
- ChatGPT Prompting Guide
- Malaria Atlas Project - For real prevalence data