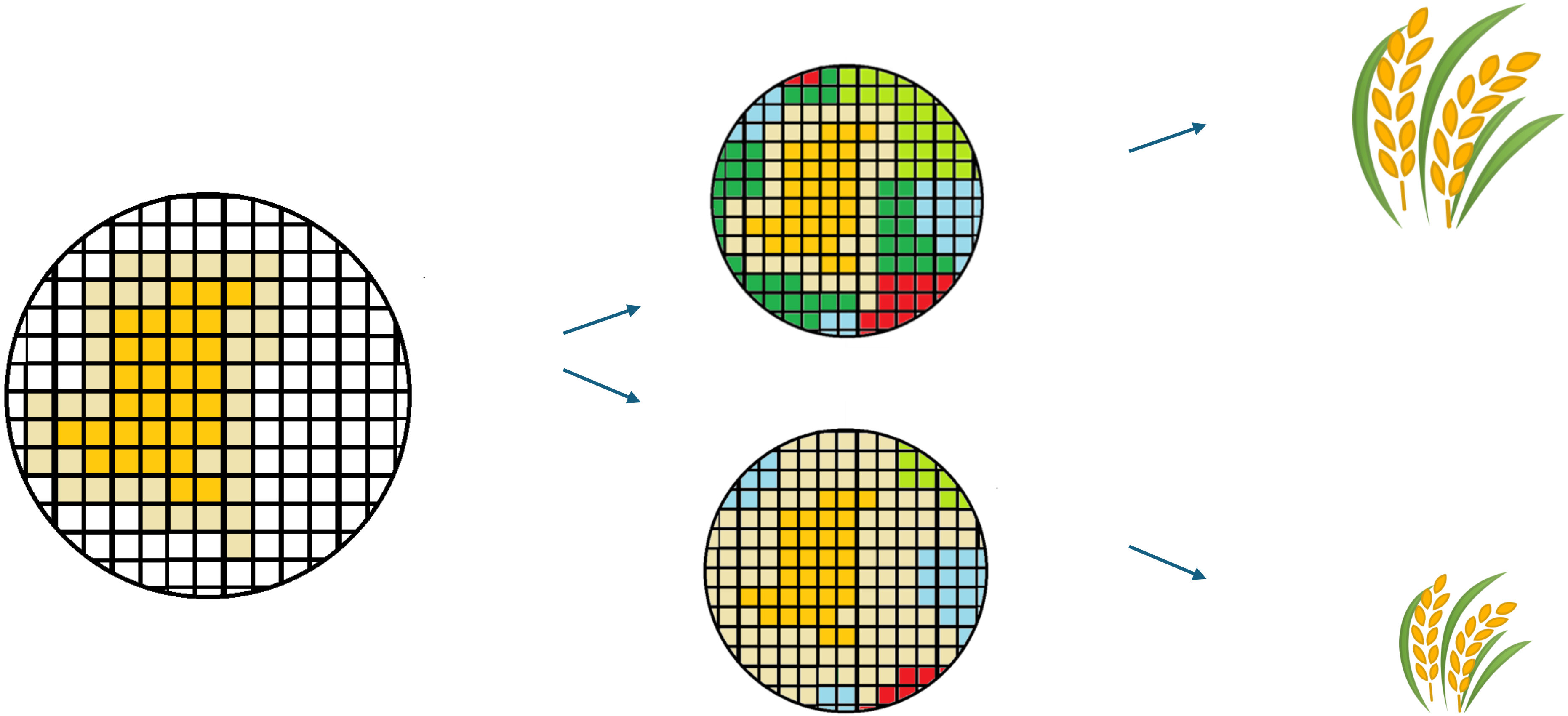
How to use Google Earth Engine to calculate landscape metrics from a set of coordinates

Workshop
organized by Elizaveta Shcherbinina
supervised by Elina Takola

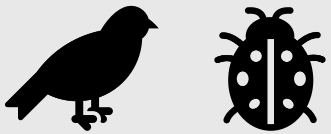
UFZ Leipzig, CLE Department
12.11.2024

“How does the heterogeneity of surrounding landscapes affect the crop yield?”

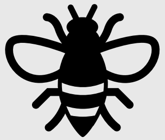


Supervised by Elina Takola

Biocontrol

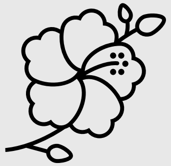


Pollination



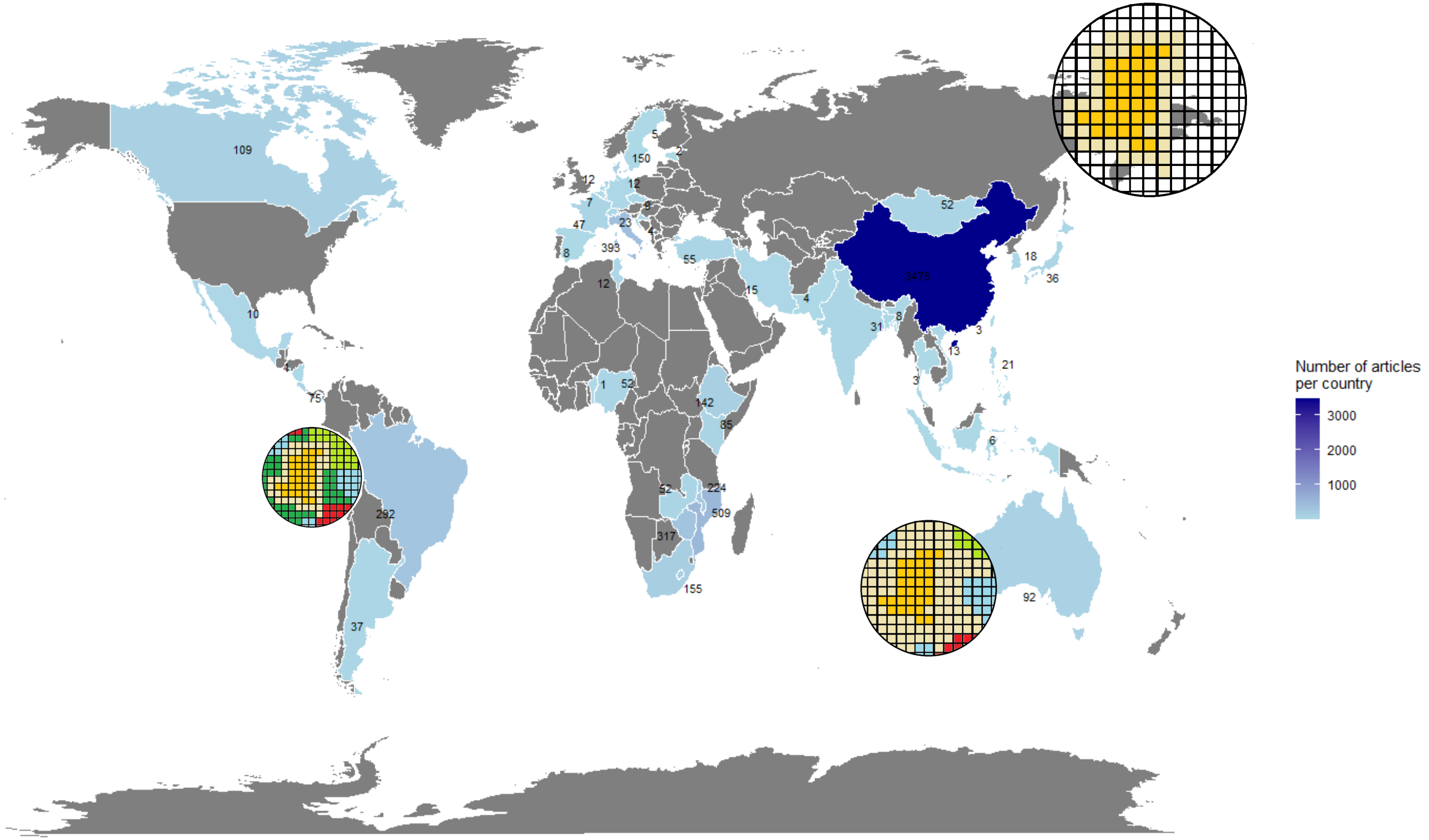
Protection

Nutrient Cycling

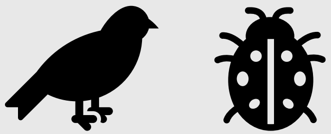


“How does the heterogeneity of surrounding landscapes affect the crop yield?”

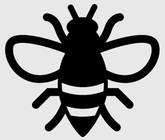
dataset: 20241027_data_yield



Biocontrol

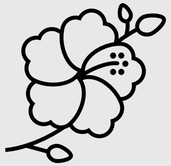


Pollination



Protection

Nutrient Cycling



Supervised by Elina Takola

ArcGIS

QGIS

R

+ GEE is Open Access
+ GEE doesn't need installation
+ GEE code runs on a server (not on your PC)

★ can be connected to R

QGIS

ArcGIS

+ GEE is Open Access
+ GEE doesn't need installation
+ GEE code runs on a server (not on your PC)

Large datasets on a small scale

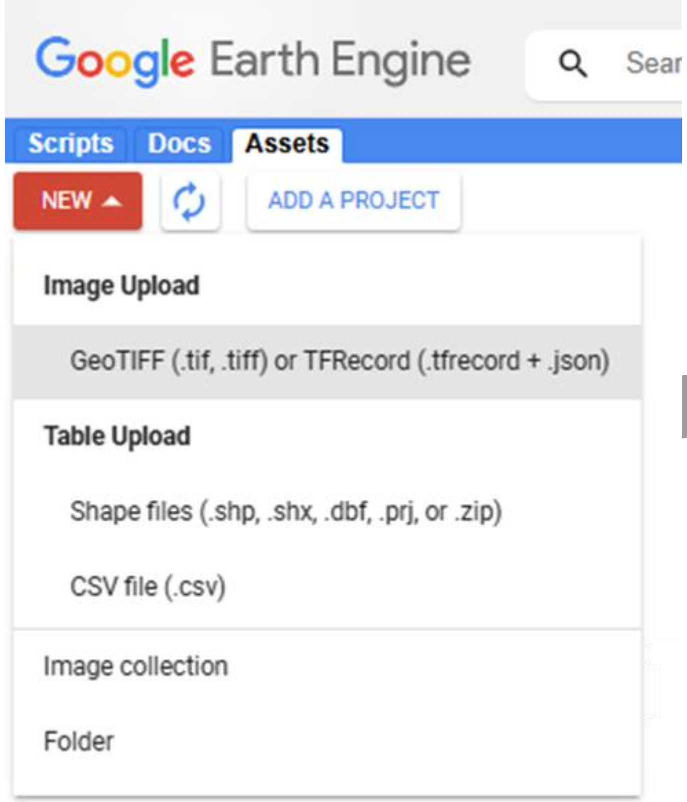
★ can be connected to R

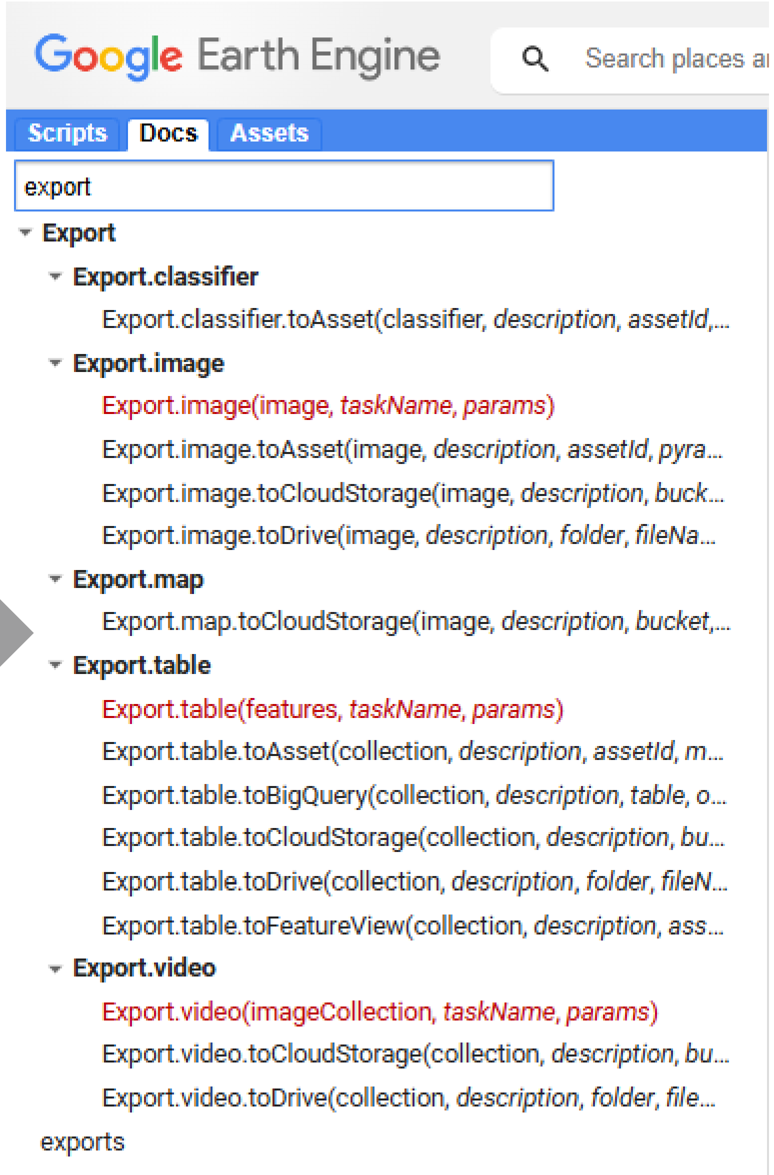
R

QGIS

ArcGIS

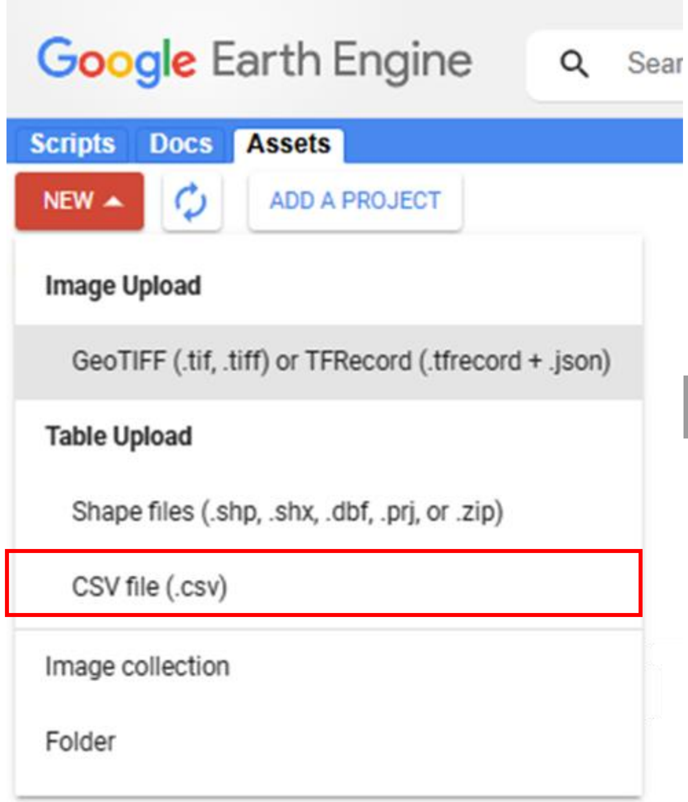
Upload

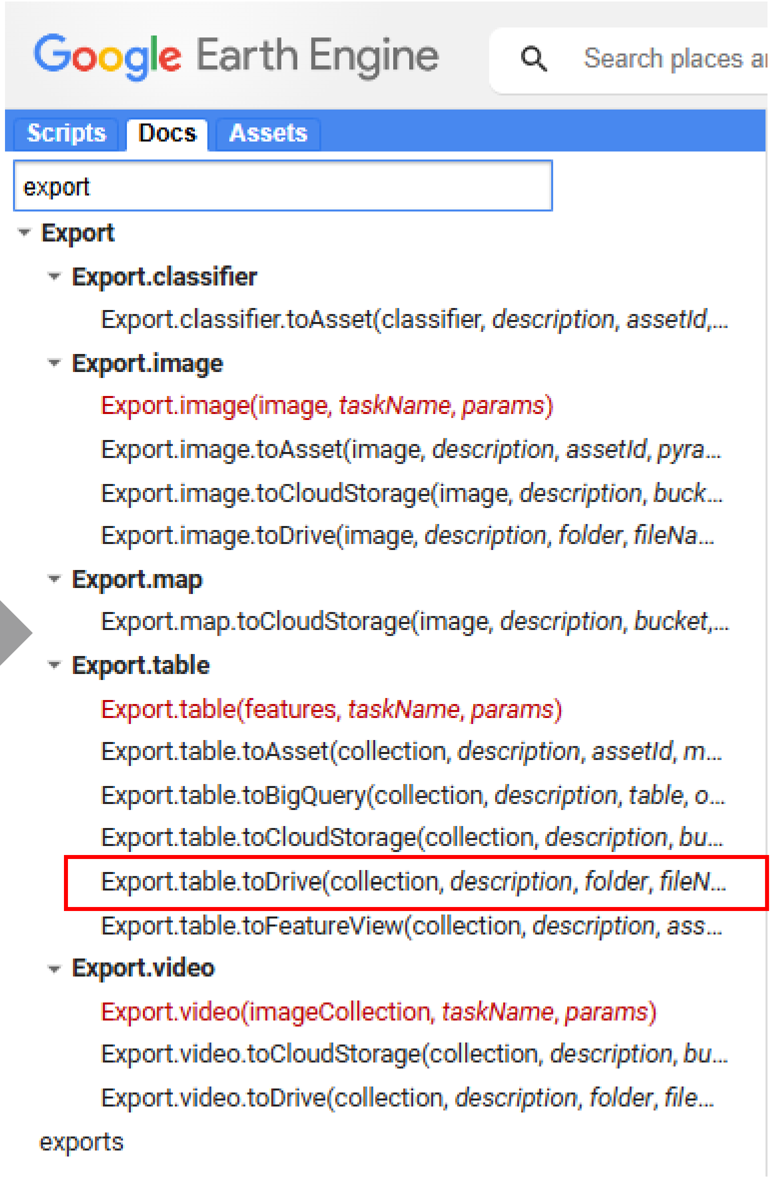




Output

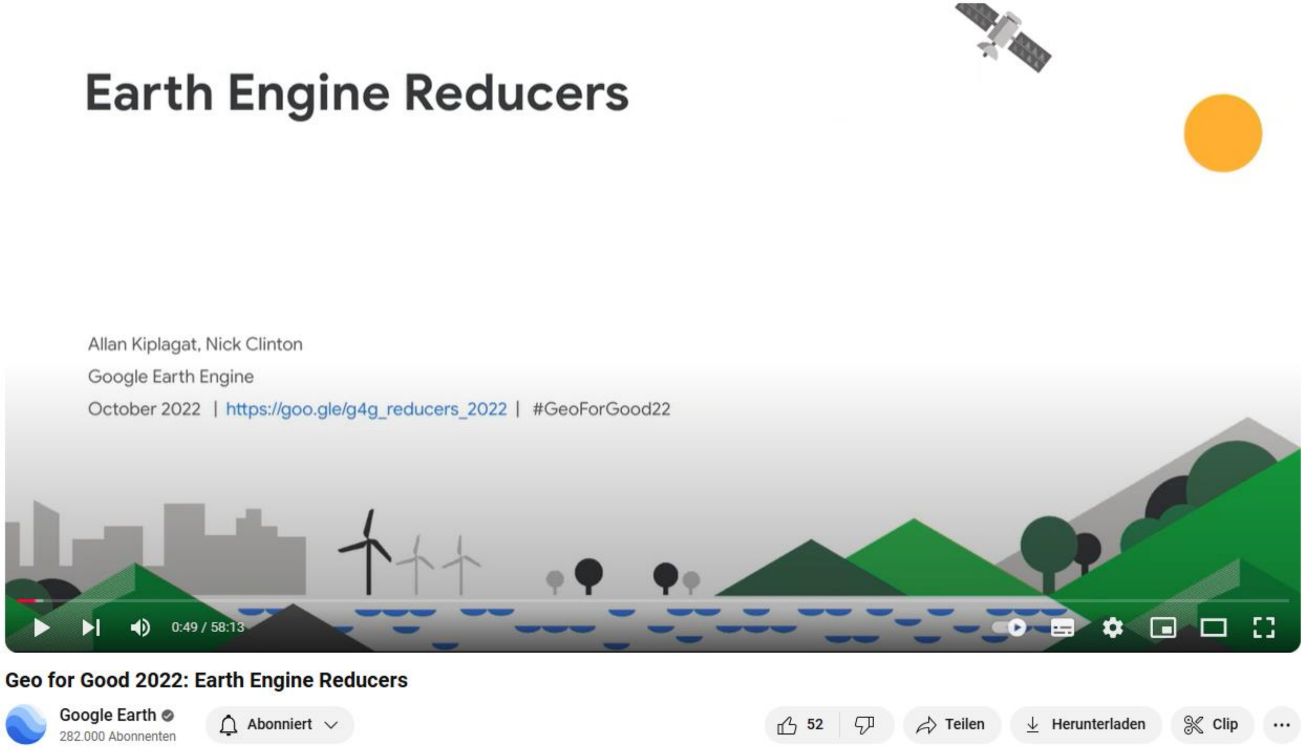
Upload



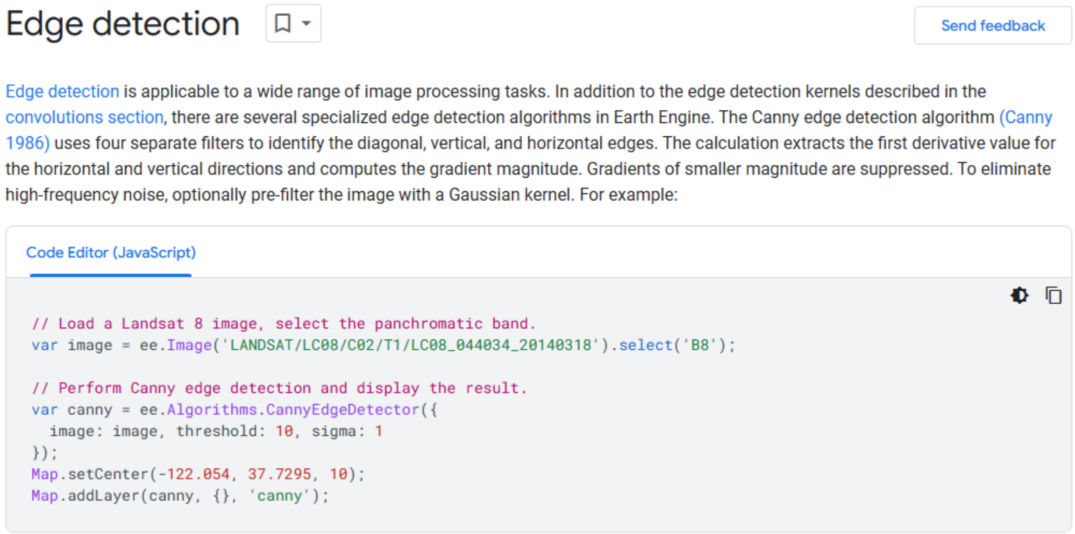


Output

Additional Help and Ressources



- [Google Earth Engine Developers]()
- [Google Earth Engine Youtube Channel]()
- [Spatial Thoughts]()
- Samapriya Roy [awesome-gee-community-catalog]()



Edge detection

Send feedback

Edge detection is applicable to a wide range of image processing tasks. In addition to the edge detection kernels described in the convolutions section, there are several specialized edge detection algorithms in Earth Engine. The Canny edge detection algorithm (Canny 1986) uses four separate filters to identify the diagonal, vertical, and horizontal edges. The calculation extracts the first derivative value for the horizontal and vertical directions and computes the gradient magnitude. Gradients of smaller magnitude are suppressed. To eliminate high-frequency noise, optionally pre-filter the image with a Gaussian kernel. For example:

Code Editor (JavaScript)

```
// Load a Landsat 8 image, select the panchromatic band.
var image = ee.Image('LANDSAT/LC08/C02/T1/LC08_044034_20140318').select('B8');

// Perform Canny edge detection and display the result.
var canny = ee.Algorithms.CannyEdgeDetector({
  image: image, threshold: 10, sigma: 1
});
Map.setCenter(-122.054, 37.7295, 10);
Map.addLayer(canny, {}, 'canny');
```

Landscape metrics calculation in GEE

Landscape metrics

- **Composition**
  - Proportional Abundance of each Class
  - Richness
  - Evenness
  - Diversity

- **Spatial configuration**
  - Patch area and edge
  - Patch shape complexity
  - Core Area
  - Contrast
  - Aggregation
  - Subdivision
  - Isolation

https://fragstats.org/index.php/background/landscape-metrics

Landscape metrics calculation in GEE

Landscape metrics

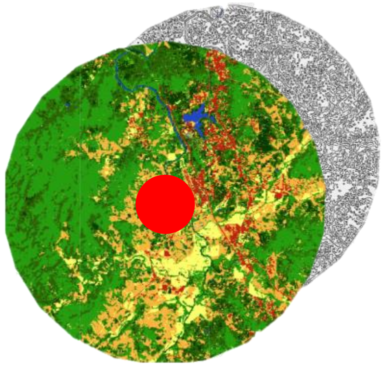
- **Composition**
  - **Proportional Abundance of each Class**
  - **Richness**
  - **Evenness**
  - **Diversity**

- **Spatial configuration**
  - **Patch area and edge**
  - Patch shape complexity
  - Core Area
  - Contrast
  - Aggregation
  - Subdivision
  - Isolation

(lat ; long)

Land Cover Map

Area & Perimeter

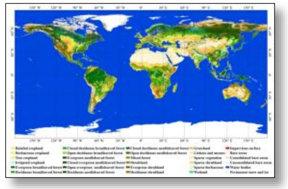


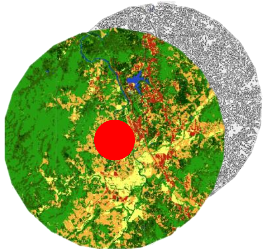
https://fragstats.org/index.php/background/landscape-metrics

Landscape metrics calculation in GEE

Ingredients:

- ❑ Landcover map
- ❑ Points with coordinates
- ❑ Buffer size





1. Upload the data with valid coordinates
2. Import your data and necessary maps
3. Filter your data points by specific year
4. Create buffers around your data points
5. Export as a csv file

Landscape metrics calculation in GEE

gee-community-catalog.org/projects/glc_fcs/

Masterschool Data... Hadley Wickham Carbon | Create and...

For updates follow @samapriya on LinkedIn GitHub Substack | Try Our AI-Enhanced Search | Support Sponsor

awesome-gee-community-catalog | Search | GitHub 3.0.0 783 125

Home | Getting Started | Data Catalog | Tutorials | Changelog | How to Cite | Insiders | Get Involved | Blogs | Advanced Search

Data Catalog

- Global Mangrove Canopy Height Maps Derived from TanDEM-X
- Randolph Glacial Inventory
- Finer Resolution Observation and Monitoring of Global Land Cover 10m (FROM-GLC10)
- ESRI 2020 Global Land Use Land Cover from Sentinel-2
- ESRI 10m Annual Land Cover (2017-2023)
- GlobCover Global Land Cover
- GLC_FCS30D Global 30-meter Land Cover Change Dataset (1985-2022)
- ESA WorldCover 10 m 2020 V100 InputQuality
- Daylight Map Distribution map data
- GLANCE Global Landcover Training dataset
- Global Land Cover Estimation (GLanCE)
- Global Impervious Surface

GLC_FCS30D Global 30-meter Land Cover Change Dataset (1985-2022)

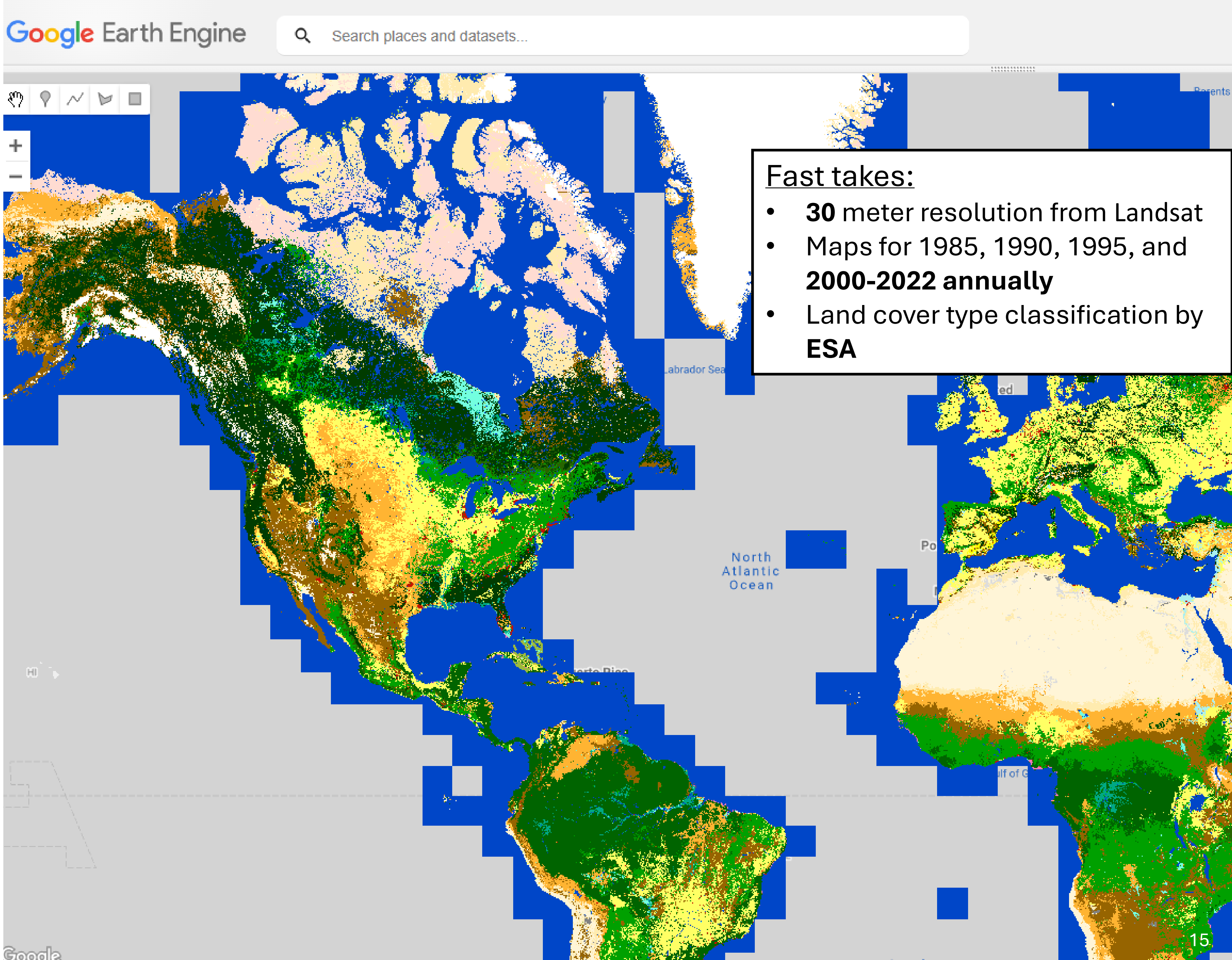
The GLC_FCS30D dataset represents a pioneering advancement in global land-cover monitoring, offering comprehensive insights into land cover dynamics at a 30-meter resolution spanning the period from 1985 to 2022. Developed using continuous change detection methods and leveraging the extensive Landsat imagery archives within the Google Earth Engine platform, GLC_FCS30D comprises 35 land-cover subcategories with 26 time steps, updated every five years prior to 2000 and annually thereafter. Through a rigorous refinement process, including spatiotemporal classification and temporal-consistency optimization, the dataset achieves high-confidence accuracy, validated with over 84,000 global samples and achieving an overall accuracy of 80.88%. Notably, GLC_FCS30D elucidates significant trends, revealing forest and cropland variations as dominant drivers of global land cover change over the past 37 years, with a net loss of approximately 2.5 million km^2 of forests and a net gain of around 1.3 million km^2 in cropland area. With its diverse classification system, high spatial resolution, and extensive temporal coverage, GLC_FCS30D serves as a valuable resource for climate change research and sustainable development analysis. Access the dataset here.

Expand to show Land Cover classes, RGB values and hex codes

Table of contents

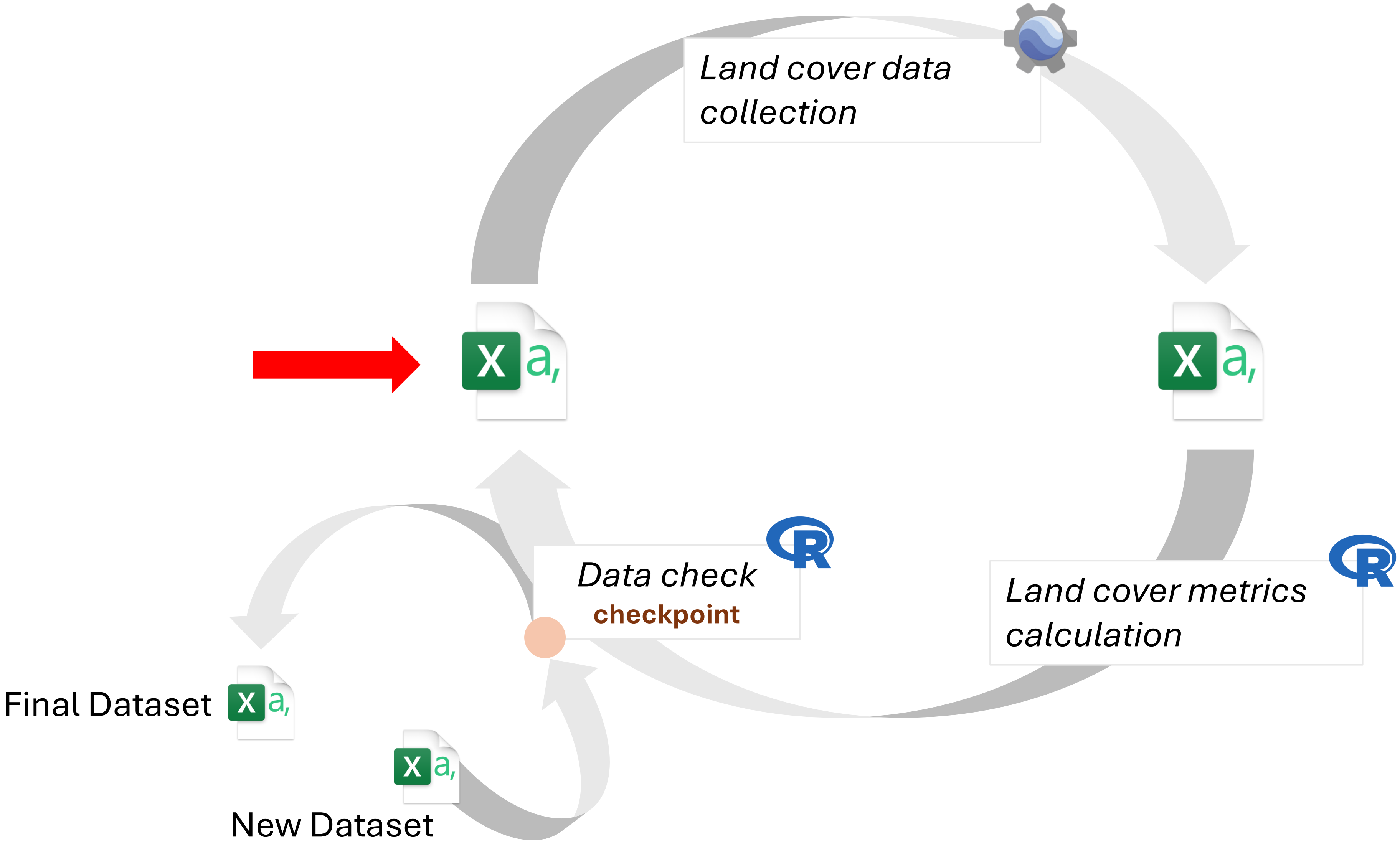
GLC FCS Classes

- Rainfed cropland
- Herbaceous cover cropland
- Tree or shrub cover (Orchard) cropland
- Irrigated cropland
- Open evergreen broadleaved forest
- Closed evergreen broadleaved forest
- Open deciduous broadleaved forest (0.15<fc<0.4)
- Closed deciduous broadleaved forest (fc>0.4)
- Open evergreen needle-leaved forest (0.15< fc <0.4)
- Closed evergreen needle-leaved forest (fc >0.4)
- Open deciduous needle-leaved forest (0.15< fc <0.4)
- Closed deciduous needle-leaved forest (fc >0.4)
- Open mixed leaf forest (broadleaved and needle-leaved)
- Closed mixed leaf forest (broadleaved and needle-leaved)
- Shrubland
- Evergreen shrubland
- Deciduous shrubland
- Grassland
- Lichens and mosses
- Sparse vegetation (fc<0.15)
- Sparse shrubland (fc<0.15)
- Sparse herbaceous (fc<0.15)
- Swamp
- Marsh
- Flooded flat
- Saline
- Mangrove
- Salt marsh
- Tidal flat
- Impervious surfaces = buildings, pavements
- Bare areas
- Consolidated bare areas
- Unconsolidated bare areas
- Water body
- Permanent ice and snow
- Filled value



Fast takes:

- **30** meter resolution from Landsat
- Maps for 1985, 1990, 1995, and **2000-2022 annually**
- Land cover type classification by **ESA**


Land cover data collection
Land cover metrics calculation
Data check
checkpoint
Final Dataset
New Dataset

Functions

funcitonName parameter1, parameter2

```
function bufferPoints(radius, bounds) {
  return function(pt) {
    pt = ee.Feature(pt);
    var bufferedGeom = bounds ? pt.buffer(radius).bounds() : pt.buffer(radius);
    return ee.Feature(bufferedGeom).copyProperties(pt);
  };
}
```

Usage example:

```
var bufferedPoints = filteredPoints.map(bufferPoints(5000));
```

Import Datasets

Load landcover data as Image Collections for two periods (1985-1995 (**five_year**)and 2000-2022 (**annual**)) and import the working dataset.

```
// (1) Import Image Collections and the points dataset
var points = ee.FeatureCollection("projects/ee-elizaveta/assets/20241010_data_yield");
var five_year = ee.ImageCollection("projects/sat-io/open-datasets/GLC-FCS30D/five-years-map");
var annual = ee.ImageCollection("projects/sat-io/open-datasets/GLC-FCS30D/annual");
```

Filter Points by Year

Filter the points dataset by specific years, creating subsets for each year of interest (1985, 1990, 1995).

```
// (2) Filter points by year
var years = [1985, 1990, 1995];
var filteredPoints = {};
years.forEach(function(year) {
  filteredPoints[year] = points.filter(ee.Filter.eq('landcover_map_year', year));
});
```

var points
var five_year
var annual
function bufferPoints

Buffer Points

Define a function to create buffered areas around each point (with customizable radius (*radius*) and shape (*bounds*)).

```
// (3) Function to buffer the points
function bufferPoints(radius, bounds) {
  return function(pt) {
    pt = ee.Feature(pt);
    var bufferedGeom = bounds ? pt.buffer(radius).bounds() : pt.buffer(radius);
    return ee.Feature(bufferedGeom).copyProperties(pt);
  };
}
```

```
// (4) Edge Length and Area Calculation
var classes = [10, 11, 12, 20, 51, 52, 61, 62, 71, 72, 81, 82, 91, 92, 120, 121, 122,
               130, 140, 150, 152, 153, 181, 182, 183, 184, 185, 186, 187, 190, 200,
               201, 202, 210, 220, 0];

```

```
// (4.1) Select images
var images = {
  1985: five_year.mosaic().select('b1'),
  1990: five_year.mosaic().select('b2'),
  1995: five_year.mosaic().select('b3')
};
```

```
// (4.2) Apply Canny edge detection AFTER buffering
function detectEdges(image, bufferedGeometry) {
  return ee.Algorithms.CannyEdgeDetector({
    image: image.clip(bufferedGeometry),
    threshold: 0.7,
    sigma: 1
  }).selfMask();
}
```

ee.Algorithms.CannyEdgeDetector(image, threshold, *sigma*)

Applies the Canny edge detection algorithm to an image. The output is an image whose bands have the same names as the input bands, and in which non-zero values indicate edges, and the magnitude of the value is the gradient magnitude.

Arguments:

- **image (Image):**
  The image on which to apply edge detection.
- **threshold (Float):**
  Threshold value. The pixel is only considered for edge detection if the gradient magnitude is higher than this threshold.
- ***sigma (Float, default: 1):***
  Sigma value for a gaussian filter applied before edge detection. 0 means apply no filtering.

Returns: Image

CLOSE

Edge Detection and Landcover Metrics

New Variable: Numeric values of the classes used in your land cover map

New Variable: Images (= bands from your image collection) you want to work with

New Function: apply Canny Edge Detection from ee.Algorithms to identify boundaries within each buffered area for different landcover classes

```
// (4.3) Function to calculate edge length and area for each class
// (4.3.1) Edge calculation:
function calculateMetrics(image, edges, geometry, classValue) {
  var classMask = image.eq(classValue);
  var classEdges = edges.updateMask(classMask);
  var edgeLength = classEdges.reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: geometry,
    scale: 5,
    maxPixels: 1e29
  }).get(image.bandNames().get(0));

// (4.3.2.) Area calculation
  var areaImage = ee.Image.pixelArea().updateMask(classMask);
  var area = areaImage.reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: geometry,
    scale: 5,
    maxPixels: 1e29
  }).get('area');

  return ee.Dictionary({class: classValue, length_m: edgeLength, area_m2: area});
}
```

Edge Detection and Landcover Metrics

var points
var five_year
var annual
function bufferPoints
var classes
var images
function detectEdges

New Function: For each landcover class, calculate the edge length and area within the buffer using the Canny edge detection algorithm and pixel area computations.

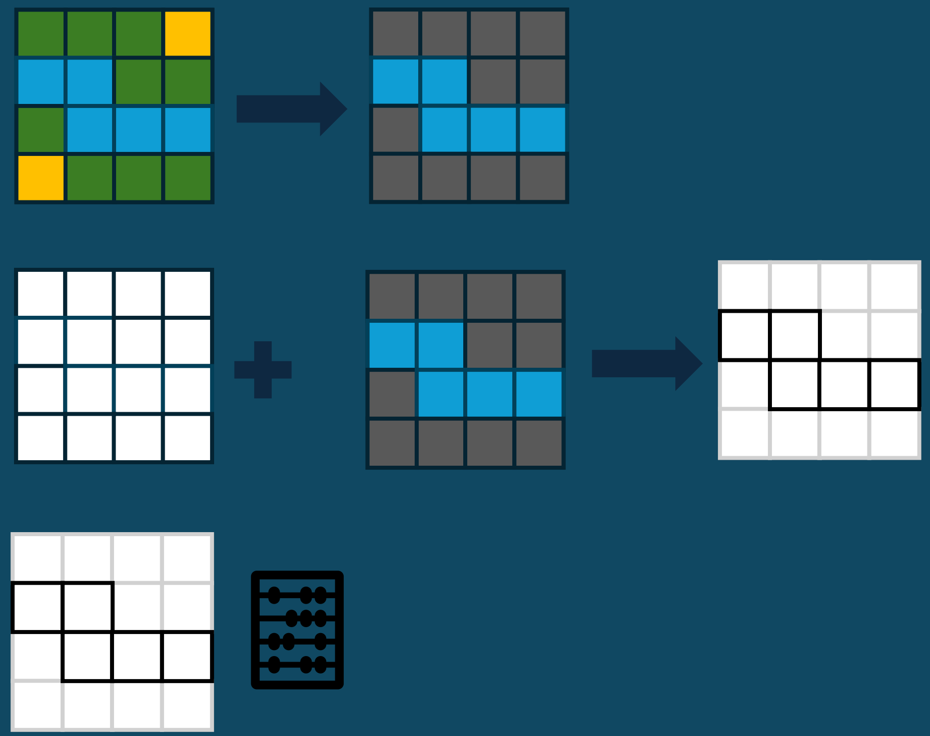
Edge Detection

```javascript
// (4.3) Function to calculate edge length and area for each class
// (4.3.1) Edge calculation:
function calculateMetrics(image, edges, geometry, classValue) {
  var classMask = image.eq(classValue);
```

```javascript
  var classEdges = edges.updateMask(classMask);
```

```javascript
  var edgeLength = classEdges.reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: geometry,
    scale: 5,
    maxPixels: 1e29
  }).get(image.bandNames().get(0));
```

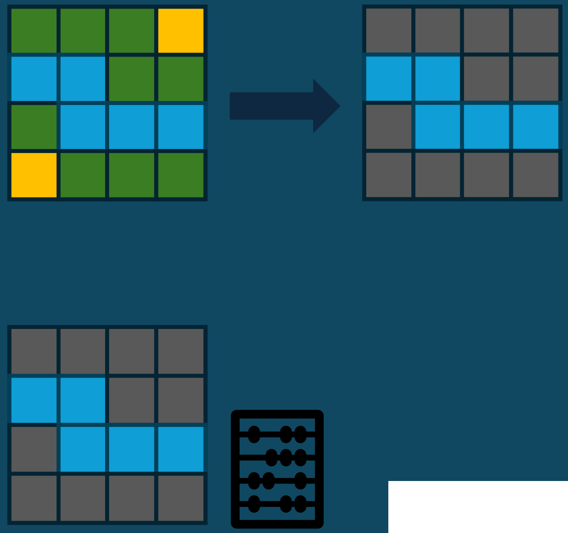
New Function: For each landcover class, calculate the edge length and area within the buffer using the Canny edge detection algorithm and pixel area computations.



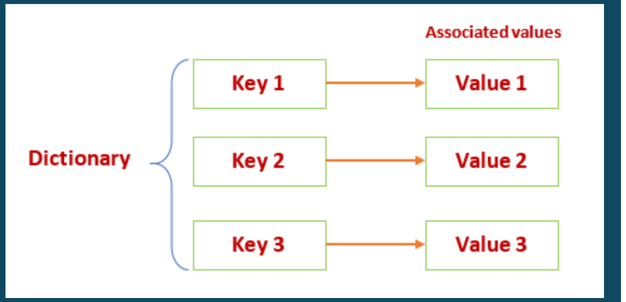
Edge Detection & Area Calculation

```
// (4.3) Function to calculate edge length and area for each class
// (4.3.1) Edge calculation:
function calculateMetrics(image, edges, geometry, classValue) {
  var classMask = image.eq(classValue);
  var classEdges = edges.updateMask(classMask);
  var edgeLength = classEdges.reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: geometry,
    scale: 5,
    maxPixels: 1e29
  }).get(image.bandNames().get(0));

// (4.3.2.) Area calculation
  var areaImage = ee.Image.pixelArea().updateMask(classMask);
  var area = areaImage.reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: geometry,
    scale: 5,
    maxPixels: 1e29
  }).get('area');

  return ee.Dictionary({class: classValue, length_m: edgeLength, area_m2: area});
}
```

New Function: For each landcover class, calculate the edge length and area within the buffer using the Canny edge detection algorithm and pixel area computations.



Output: **Dictionary**



```javascript
// (4.4) Process edge length and area for points in a given year
function processMetricsForYear(filteredPoints, image, year) {
  var bufferedPoints = filteredPoints.map(bufferPoints(5000));
  var results = bufferedPoints.map(function(point) {
    var bufferGeom = point.geometry();
    var edges = detectEdges(image, bufferGeom);
    var metrics = ee.List(classes.map(function(classValue) {
      return calculateMetrics(image, edges, bufferGeom, classValue);
    }));
    return point.set('metrics', metrics);
  });
  return results;
}
```

Edge Detection and Landcover Metrics

New Function: Apply the calculateMetrics function to the points features. For that filter the points by year and create a buffer in which the calculation should happen.

```
// (4.4) Process edge length and area for points in a given year
function processMetricsForYear(filteredPoints, image, year) {
  var bufferedPoints = filteredPoints.map(bufferPoints(5000));
  var results = bufferedPoints.map(function(point) {
    var bufferGeom = point.geometry();
    var edges = detectEdges(image, bufferGeom);
    var metrics = ee.List(classes.map(function(classValue) {
      return calculateMetrics(image, edges, bufferGeom, classValue);
    }));
    return point.set('metrics', metrics);
  });
  return results;
}
```

var points
var five_year
var annual
function bufferPoints
var classes
var images
function detectEdges
function calculateMetrcs

New Function: Apply the calculateMetrics function to the points features. For that filter the points by year and create a buffer in which the calculation should happen.

```
260 // (4.4) Process edge length and area for points in a given year
261 function processMetricsForYear(filteredPoints, image, year) {
262   var bufferedPoints = filteredPoints.map(bufferPoints(5000));
```

```
263   var results = bufferedPoints.map(function(point) {
264     var bufferGeom = point.geometry();
265     var edges = detectEdges(image, bufferGeom);
266     var metrics = ee.List(classes.map(function(classValue) {
267       return calculateMetrics(image, edges, bufferGeom, classValue);
268     }));
269     return point.set('metrics', metrics);
270   });
271   return results;
272 }
```

```
274 // (4.5) Unpack the metrics
275 function unpackMetrics(feature) {
276   var metrics = ee.List(feature.get('metrics'));
277   var newFeatures = metrics.map(function(metric) {
278     metric = ee.Dictionary(metric);
279     return feature.copyProperties(feature).set({
280       'class': metric.get('class'),
281       'length_m': metric.get('length_m'),
282       'area_m2': metric.get('area_m2')
283     });
284   });
285   return ee.FeatureCollection(newFeatures);
286 }
```

New Function: Apply the calculateMetrics function to the points features. For that filter the points by year and create a buffer in which the calculation should happen.

- 262: Create buffer around data points

- 264: Save the point's geometry
- 265: Detect the edges within the buffered image
- 266-268: Apply the calculateMetrics function
- 269: Add the created "metrics" feature to the initial point feature

- Unpack your dictionary

```
// (4.6) Apply function to process metrics (edge length and area) for each year

years.forEach(function(year) {

  var metricsResults = processMetricsForYear(filteredPoints[year], images[year], year);
  var unpackedFeatureCollection = ee.FeatureCollection(metricsResults).map(unpackMetrics).flatten();

  Export.table.toDrive({
    collection: unpackedFeatureCollection,
    description: 'lm_metrics_5000m_' + year,
    folder: "GEE metrics 5000 m",
    fileFormat: 'CSV'
  });
});
```

New Function: Apply the calculateMetrics function to the points features. For that filter the points by year and create a buffer in which the calculation should happen.

map

buffer

processing

unpacking

exporting

Inspector Console **Tasks**

Search or cancel multiple tasks in the Task Manager or try the Tasks Page in the Cloud Console

UNSUBMITTED TASKS

- lm_metrics_5000m_1985 RUN
- lm_metrics_5000m_1990 RUN
- lm_metrics_5000m_1995 RUN
- lm_metrics_5000m_2000 RUN
- lm_metrics_5000m_2001 RUN
- lm_metrics_5000m_2002 RUN
- lm_metrics_5000m_2003 RUN
- lm_metrics_5000m_2004 RUN
- lm_metrics_5000m_2005 RUN
- lm_metrics_5000m_2006 RUN
- lm_metrics_5000m_2007 RUN
- lm_metrics_5000m_2008 RUN
- lm_metrics_5000m_2009 RUN
- lm_metrics_5000m_2010 RUN
- lm_metrics_5000m_2011
- lm_metrics_5000m_2012
- lm_metrics_5000m_2013
- lm_metrics_5000m_2014
- lm_metrics_5000m_2015

Task: Initiate table export

Task name (no spaces) *
lm_metrics_5000m_1985

DRIVE | CLOUD STORAGE | EE ASSET | FEATURE VIEW ASSET | BIGQUERY

Drive folder
GEE metrics 5000 m

Filename *
lm_metrics_5000m_1985

File format *
CSV

CANCEL RUN

New Function: Apply the calculateMetrics function to the points features. For that filter the points by year and create a buffer in which the calculation should happen.

map

buffer

processing

unpacking

exporting

```
// For annual mosaic (2000 onwards)
for (var i = 1; i <= 16; i++) {
  var year = 1999 + i;  // starts at year 2000
  var image = annual.mosaic().select("b" + i);
  var filteredPoints = points.filter(ee.Filter.eq('landcover_map_year', year));

var  edge = ee.Algorithms.CannyEdgeDetector({
    image: image,
    threshold: 0.7,
    sigma: 1
  }).selfMask();

  var metricsResults = processMetricsForYear(filteredPoints, image, year);
  var unpackedFeatureCollection = ee.FeatureCollection(metricsResults).map(unpackMetrics).flatten();

  Map.addLayer(edge, {palette: ['white'], min: 0, max: 1}, 'Edges for ' + year);

  Export.table.toDrive({
    collection: unpackedFeatureCollection,
    description: 'lm_metrics_5000m_' + year,
    folder: "GEE metrics 5000 m",
    fileFormat: 'CSV'
  });
```

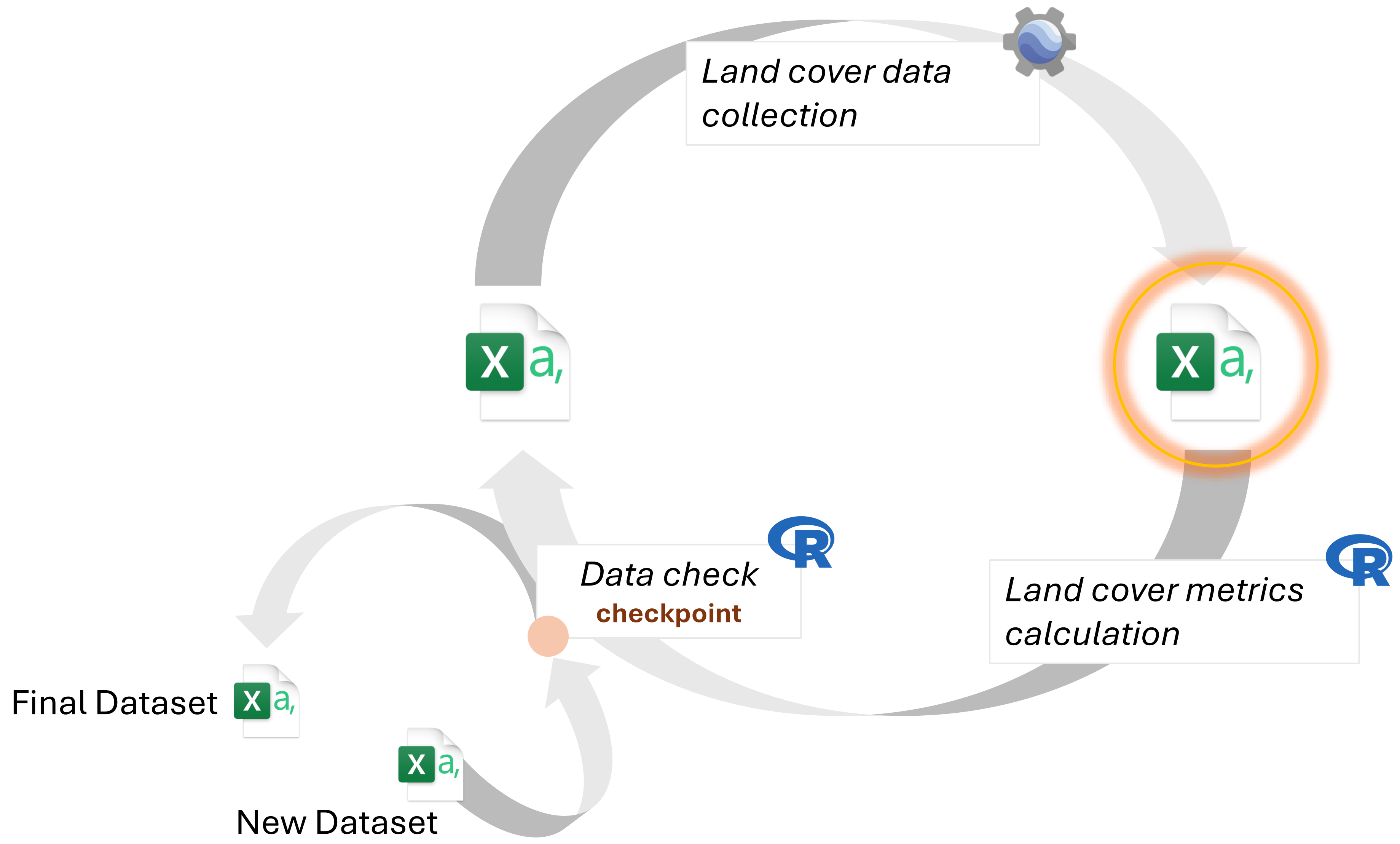
Repeat for annual data set

```
years.forEach(function(year) {
  filteredPoints[year] = points.filter(ee.Filter.eq('landcover_map_year', year)); // fill in w


// (3) Function to buffer the points
function bufferPoints(radius, bounds) { // here in the arguments you can define your radius (r
  return function(pt) {
    pt = ee.Feature(pt);
    var bufferedGeom = bounds ? pt.buffer(radius).bounds() : pt.buffer(radius);
    return ee.Feature(bufferedGeom).copyProperties(pt);
  };
}

// (4) Edge Length and Area Calculation
var classes = [10, 11, 12, 20, 51, 52, 61, 62, 71, 72, 81, 82, 91, 92, 120, 121, 122,
               130, 140, 150, 152, 153, 181, 182, 183, 184, 185, 186, 187, 190, 200,
               201, 202, 210, 220, 0]; // a list of all land cover classes

// (4.1) Select images
var images = {
  1985: five_year.mosaic().select('b1'),
  1990: five_year.mosaic().select('b2'),
  1995: five_year.mosaic().select('b3')
};

// (4.2) Apply Canny edge detection AFTER buffering
function detectEdges(image, bufferedGeometry) {
  return ee.Algorithms.CannyEdgeDetector({
    image: image.clip(bufferedGeometry),  // Detect edges inside the buffer
    threshold: 0.7,
    sigma: 1
  }).selfMask();
}


// (4.3) Function to calculate edge length and area for each class
// (4.3.1) Edge calculation:
function calculateMetrics(image, edges, geometry, classValue) {
  var classMask = image.eq(classValue);
  var classEdges = edges.updateMask(classMask);
  var edgeLength = classEdges.reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: geometry,
    scale: 5,
    maxPixels: 1e29
  }).get(image.bandNames().get(0));

// (4.3.2.) Area calculation
  var areaImage = ee.Image.pixelArea().updateMask(classMask);
  var area = areaImage.reduceRegion({
    reducer: ee.Reducer.sum(),
    geometry: geometry,
    scale: 5,
    maxPixels: 1e29
  }).get('area');

  return ee.Dictionary({class: classValue, length_m: edgeLength, area_m2: area});
}

// (4.4) Process edge length and area for points in a given year
function processMetricsForYear(filteredPoints, image, year) {
  var bufferedPoints = filteredPoints.map(bufferPoints(5000));
  var results = bufferedPoints.map(function(point) {
    var bufferGeom = point.geometry();
```

* Your final toolkit *

var points
var five_year
var annual
function bufferPoints
var classes
var images
function detectEdges
function calculateMetrcs
funciton processMetrcisForYear
function unpackMetrics

Questions?


Land cover data collection
Data check
checkpoint
Land cover metrics calculation
Final Dataset
New Dataset

Merge the csv files

```r
# 11. Merge the GEE outputs  ---------------------------------------------


#### 1000 m #####
lm_metrics_1000m_1985 <- read.csv("my_computer_files/GEE metrics 1000 m/lm_metrics_1000m_1985.csv", colClasses = "character")
lm_metrics_1000m_1990 <- read.csv("my_computer_files/GEE metrics 1000 m/lm_metrics_1000m_1990.csv", colClasses = "character")
lm_metrics_1000m_1995 <- read.csv("my_computer_files/GEE metrics 1000 m/lm_metrics_1000m_1995.csv", colClasses = "character")

# Initialize an empty list to store the data frames
lm_metrics_1000m <- list()

for (year in c(2000:2015)) {
  file_path <- paste0("my_computer_files/GEE metrics 1000 m/lm_metrics_1000m_", year, ".csv")

  lm_metrics_1000m[[year]] <- read.csv(file_path, colClasses = "character")
}

# Combine all data frames into one
lm_metrics_full_1000m <- bind_rows(lm_metrics_1000m)

lm_metrics_full_1000m <-
  bind_rows(lm_metrics_full_1000m, lm_metrics_1000m_1985,
            lm_metrics_1000m_1990, lm_metrics_1000m_1995)



# prepare for future merge with data_pr dataset:
lm_metrics_full_1000m <- lm_metrics_full_1000m[,colSums(is.na(lm_metrics_full_1000m))<nrow(lm_metrics_full_1000m)]

lm_metrics_full_1000m <-
  lm_metrics_full_1000m %>%
  mutate(buffer_radius_m = 1000)

rm(lm_metrics_1000m, lm_metrics_1000m_1985, lm_metrics_1000m_1990, lm_metrics_1000m_1995)
```

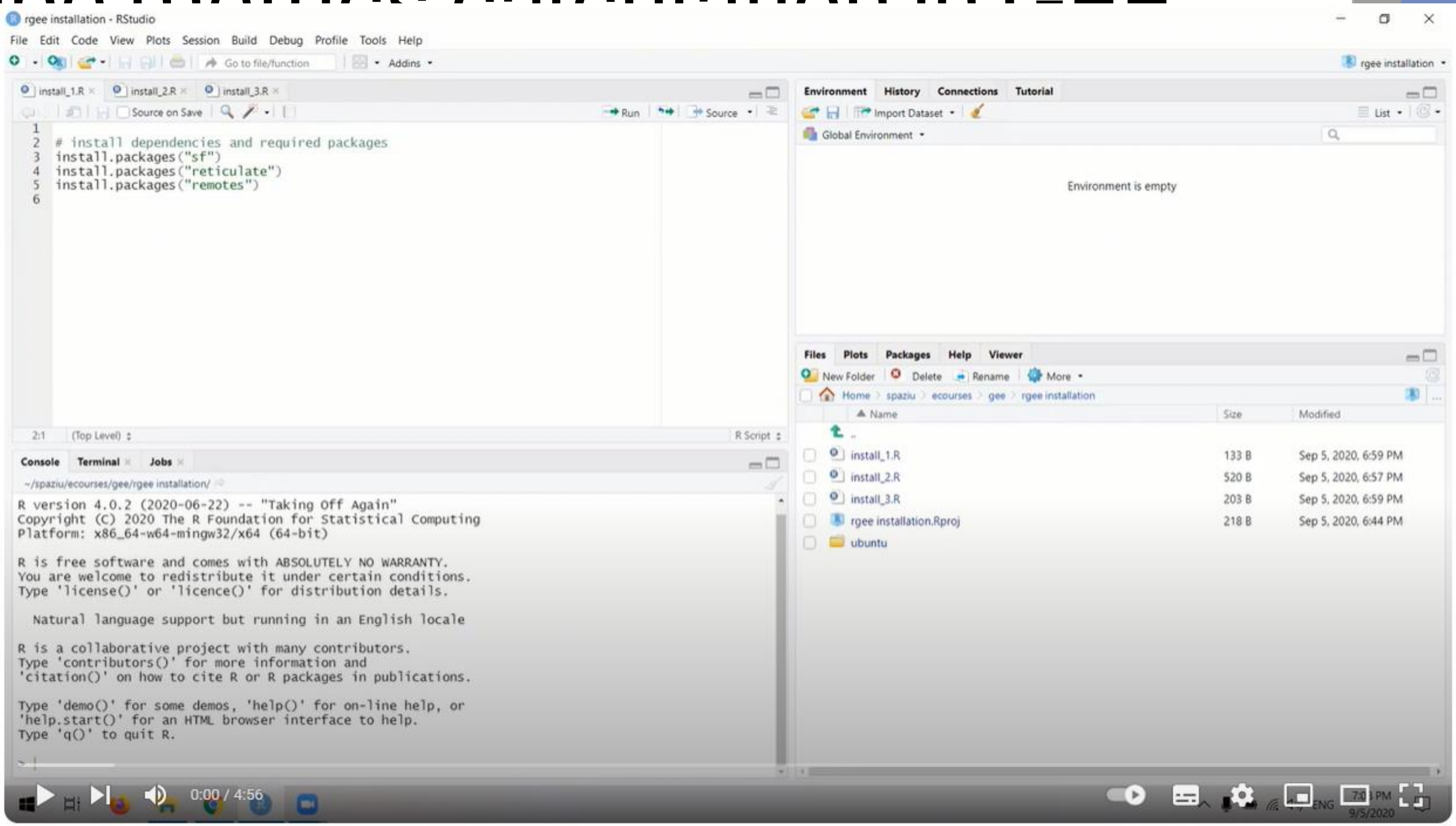
Merge the csv files

```
lm_metrics_full <- bind_rows(lm_metrics_full_1000m,
                             lm_metrics_full_2500m,
                             lm_metrics_full_5000m)

rm(lm_metrics_full_1000m, lm_metrics_full_2500m, lm_metrics_full_5000m)

lm_metrics_full_selected <-
lm_metrics_full %>%
  select(measurement_id, class, area_m2, length_m, buffer_radius_m) %>%
  rename(edgelength_m = length_m) %>%
  mutate(measurement_id = as.integer(measurement_id),
         area_m2 = as.numeric(area_m2),
         edgelength_m = as.numeric(edgelength_m),
         buffer_radius_m = as.numeric(buffer_radius_m))
```

Ready to calculate the landscape metrics ?

Landscape metrics calculation in GEE

- **Comp**
  - **Pr**
  - **Ri**
  - **Ev**
  - **Di**
- **Spatia**
  - **Pa**



How to install the R rgee package to work with Google Earth Engine from R

Data Ninjas
993 Abonnenten
Abonnieren
83 Teilen Herunterladen Clip

ster data,
his case

and GEE
makes
spatial

https://fragstats.org/index.php/background/landscape-metrics

Landscape metrics calculation in GEE

- **Composition**
  - **Proportional Abundance of each Class**
  - **Richness**
  - **Evenness**
  - **Diversity**

- **Spatial configuration**
  - **Patch area and edge**

(L7) Shannon's Evenness Index

$SHEI = \frac{-\sum_{i=1}^{m}(P_i * \ln P_i)}{\ln m}$	Pi = proportion of the landscape occupied by patch type (class) i. m = number of patch types (classes) present in the landscape, excluding the landscape border if present.
Description	SHEI equals minus the sum, across all patch types, of the proportional abundance of each patch type multiplied by that proportion, divided by the logarithm of the number of patch types. In other words, the observed Shannon's Diversity Index divided by the maximum Shannon's Diversity Index for that number of patch types. Note, Pi is based on total landscape area (A) excluding any internal background present.
Units	None
Range	0 ≤ SHEI ≤ 1 SHDI = 0 when the landscape contains only 1 patch (i.e., no diversity) and approaches 0 as the distribution of area among the different patch types becomes increasingly uneven (i.e., dominated by 1 type). SHDI = 1 when distribution of area among patch types is perfectly even (i.e., proportional abundances are the same).
Comments	Shannon's evenness index is expressed such that an even distribution of area among patch types results in maximum evenness. As such, evenness is the complement of dominance.

https://fragstats.org/index.php/background/landscape-metrics

```r
# 11.1. Calculate additional metrics ---------------------------------------

lm_metrics_full_selected_calc <-
  lm_metrics_full_selected %>%
  group_by(measurement_id, buffer_radius_m, class) %>%
  mutate(buffer_area_m2 = (pi*buffer_radius_m^2),
         proportion = area_m2/buffer_area_m2)

# for the simpsons formula the following protocoll was used :
# https://search.r-project.org/CRAN/refmans/abdiv/html/simpson.html
# however one can consider using sum(area) instead of buffer_area
lm_metrics_full_selected_calc <-
  lm_metrics_full_selected_calc %>%
  group_by(measurement_id, buffer_radius_m) %>%
  mutate(dividend = area_m2*(area_m2-1)) %>%
  mutate(simpsons_index = 1-(sum(dividend)/(buffer_area_m2*(buffer_area_m2-1)))) %>%
  mutate(species_richness = n_distinct(class)) %>%
  mutate(simpsons_evenness = 1/(simpsons_index*species_richness)) %>%
  mutate(perimeter_to_area = edgelength_m/area_m2) %>%
  select(-dividend)

lm_metrics_full_selected_calc <-
  lm_metrics_full_selected_calc %>%
  group_by(measurement_id, buffer_radius_m) %>%
  mutate(shannons_index = -(sum(if_else(proportion > 0, proportion * log(proportion), 0))))

# 12. Merge with the bigger dataset  ---------------------------------------

data_20241007 <- read.csv("C:\\Users\\lisa7\\Documents\\surrounding_landscapes\\data\\working data\\20241010_data_yield.csv")

df_20241020 <-
  left_join(data_20241007, lm_metrics_full_selected_calc, join_by("measurement_id"=="measurement_id"))
```

Personal note:
I use **tidyr**, **dplyr** and **piping!**

```r
# 11.1. Calculate additional metrics ---------------------------------------

lm_metrics_full_selected_calc <-
  lm_metrics_full_selected %>%
  group_by(measurement_id, buffer_radius_m, class) %>%
  mutate(buffer_area_m2 = (pi*buffer_radius_m^2),
         proportion = area_m2/buffer_area_m2)
```

V	W	X	D	E	F	S
measurement_id	metrics	precipitat	area_m2	author_ye	class	length_m
7859	[{area_m2=607831.45		607831.45	Posner et	10	572288.16
7859	[{area_m2=607831.45		6.7616140	Posner et	11	7709290.0
7859	[{area_m2=607831.45		0	Posner et	12	0
7859	[{area_m2=607831.45		227648.78	Posner et	20	210457.38
7859	[{area_m2=607831.45		541084.72	Posner et	51	989141.13
7859	[{area_m2=607831.45		1964.8807	Posner et	52	4471.2638
7859	[{area_m2=607831.45		71822.016	Posner et	61	130140.37
7859	[{area_m2=607831.45		2500198.2	Posner et	62	1896496.7
7859	[{area_m2=607831.45		280299.26	Posner et	71	812553.31
7859	[{area_m2=607831.45		24984.922	Posner et	72	44459.633
7859	[{area_m2=607831.45		47165.152	Posner et	81	78984.139
7859	[{area_m2=607831.45		8511.1793	Posner et	82	12684.116
7859	[{area_m2=607831.45		214863.12	Posner et	91	719615.33
7859	[{area_m2=607831.45		0	Posner et	92	0
7859	[{area_m2=607831.45		324092.68	Posner et	120	1182553.8
7859	[{area_m2=607831.45		4583.981	Posner et	121	13888.782
7859	[{area_m2=607831.45		0	Posner et	122	0
7859	[{area_m2=607831.45		1996880.9	Posner et	130	5292766.3
7859	[{area_m2=607831.45		654.50826	Posner et	140	3440.7440
7859	[{area_m2=607831.45		66565.335	Posner et	150	337555.99
7859	[{area_m2=607831.45		0	Posner et	152	0
7859	[{area_m2=607831.45		0	Posner et	153	0
7859	[{area_m2=607831.45		90825.51	Posner et	181	296343.35
7859	[{area_m2=607831.45		582776.9	Posner et	182	1171056.7
7859	[{area_m2=607831.45		1309.4799	Posner et	183	3115.4051
7859	[{area_m2=607831.45		0	Posner et	184	0
7859	[{area_m2=607831.45		0	Posner et	185	0
7859	[{area_m2=607831.45		0	Posner et	186	0
7859	[{area_m2=607831.45		0	Posner et	187	0
7859	[{area_m2=607831.45		2404828.5	Posner et	190	5177559.2
7859	[{area_m2=607831.45		7202.4402	Posner et	200	35128.528
7859	[{area_m2=607831.45		0	Posner et	201	0
7859	[{area_m2=607831.45		0	Posner et	202	0
			121064.63	Posner et	210	121075.12
			0	Posner et	220	0
7859	[{area_m2=607831.45		0	Posner et	0	0

Data set structure

```
# 11.1. Calculate additional metrics ---------------------------------------

lm_metrics_full_selected_calc <-
  lm_metrics_full_selected %>%
  group_by(measurement_id, buffer_radius_m, class) %>%
  mutate(buffer_area_m2 = (pi*buffer_radius_m^2),
         proportion = area_m2/buffer_area_m2)
```

Group your data by the unique id, buffer radius and land cover class. Then, calculate the **buffer area** as an area of a circle. Then, calculate the **proportion** of each class within the buffer.

```
# for the simpsons formula the following protocoll was used :
# https://search.r-project.org/CRAN/refmans/abdiv/html/simpson.html
# however one can consider using sum(area) instead of buffer_area
lm_metrics_full_selected_calc <-
  lm_metrics_full_selected_calc %>%
  group_by(measurement_id, buffer_radius_m) %>%
  mutate(dividend = area_m2*(area_m2-1)) %>%
  mutate(simpsons_index = 1-(sum(dividend)/(buffer_area_m2*(buffer_area_m2-1)))) %>%
  mutate(species_richness = n_distinct(class)) %>%
  mutate(simpsons_evenness = 1/(simpsons_index*species_richness)) %>%
  mutate(perimeter_to_area = edgelength_m/area_m2) %>%
  select(-dividend)
```

Manually calculate **simpsons evenness index** as the probability of selecting two individuals from different species, with replacement.

Details

For a vector of species counts x, the dominance index is defined as

$$D = \sum_i p_i^2,$$

where p_i is the species proportion, $p_i = x_i/N$, and N is the total number of counts. This is equal to the probability of selecting two individuals from the same species, with replacement. Relation to other definitions:

- Equivalent to `dominance()` in `skbio.diversity.alpha`.
- Similar to the `simpson` calculator in Mothur. They use the unbiased estimate $p_i = x_i(x_i - 1)/(N(N-1))$.

Simpson's index is defined here as $1 - D$, or the probability of selecting two individuals from different species, with replacement. Relation to other definitions:

- Equivalent to `diversity()` in `vegan` with `index = "simpson"`.
- Equivalent to `simpson()` in `skbio.diversity.alpha`.

```
# (iv) calculate the shannon diversity
library(vegan) # 'species' need to be the columns
reg.data$shannon.1000<-diversity(shannon.df, index = "shannon")
```

```
# 11.1. Calculate additional metrics ---------------------------------------

lm_metrics_full_selected_calc <-
  lm_metrics_full_selected %>%
  group_by(measurement_id, buffer_radius_m, class) %>%
  mutate(buffer_area_m2 = (pi*buffer_radius_m^2),
         proportion = area_m2/buffer_area_m2)
```

```
# for the simpsons formula the following protocoll was used :
# https://search.r-project.org/CRAN/refmans/abdiv/html/simpson.html
# however one can consider using sum(area) instead of buffer_area
lm_metrics_full_selected_calc <-
  lm_metrics_full_selected_calc %>%
  group_by(measurement_id, buffer_radius_m) %>%
  mutate(dividend = area_m2*(area_m2-1)) %>%
  mutate(simpsons_index = 1-(sum(dividend)/(buffer_area_m2*(buffer_area_m2-1)))) %>%
  mutate(species_richness = n_distinct(class)) %>%
  mutate(simpsons_evenness = 1/(simpsons_index*species_richness)) %>%
  mutate(perimeter_to_area = edgelength_m/area_m2) %>%
  select(-dividend)
```

```
lm_metrics_full_selected_calc <-
  lm_metrics_full_selected_calc %>%
  group_by(measurement_id, buffer_radius_m) %>%
  mutate(shannons_index = -(sum(if_else(proportion > 0, proportion * log(proportion), 0))))
```

```
# Using vegan to calculate Shannon Diversity
species_counts <- c(10, 20, 30, 40)
shannon_diversity_vegan <- diversity(species_counts, index="shannon")
print(shannon_diversity_vegan)
```

Group your data by the unique id, buffer radius and land cover class. Then, calculate the **buffer area** as an area of a circle. Then, calculate the **proportion** of each class within the buffer.

Manually calculate **simpsons evenness index** as the probability of selecting two individuals from different species, with replacement.

Calculate **shannons diversity index** as p is the proportion (n/N) of individuals of one particular species found (n) divided by the total number of individuals found (N), ln is the natural log, Σ is the sum of the calculations.

$$\text{Shannon Index (H)} = -\sum_{i=1}^{s} p_i \ln p_i$$

```
# 11.1. Calculate additional metrics ------------------------------------

lm_metrics_full_selected_calc <-
  lm_metrics_full_selected %>%
  group_by(measurement_id, buffer_radius_m, class) %>%
  mutate(buffer_area_m2 = (pi*buffer_radius_m^2),
         proportion = area_m2/buffer_area_m2)
```

Group your data by the unique id, buffer radius and land cover class. Then, calculate the **buffer area** as an area of a circle. Then, calculate the **proportion** of each class within the buffer.

```
# for the simpsons formula the following protocoll was used :
# https://search.r-project.org/CRAN/refmans/abdiv/html/simpson.html
# however one can consider using sum(area) instead of buffer_area
lm_metrics_full_selected_calc <-
  lm_metrics_full_selected_calc %>%
  group_by(measurement_id, buffer_radius_m) %>%
  mutate(dividend = area_m2*(area_m2-1)) %>%
  mutate(simpsons_index = 1-(sum(dividend)/(buffer_area_m2*(buffer_area_m2-1)))) %>%
  mutate(species_richness = n_distinct(class)) %>%
  mutate(simpsons_evenness = 1/(simpsons_index*species_richness)) %>%
  mutate(perimeter_to_area = edgelength_m/area_m2) %>%
  select(-dividend)
```

Manually calculate **simpsons evenness index** as the probability of selecting two individuals from different species, with replacement.

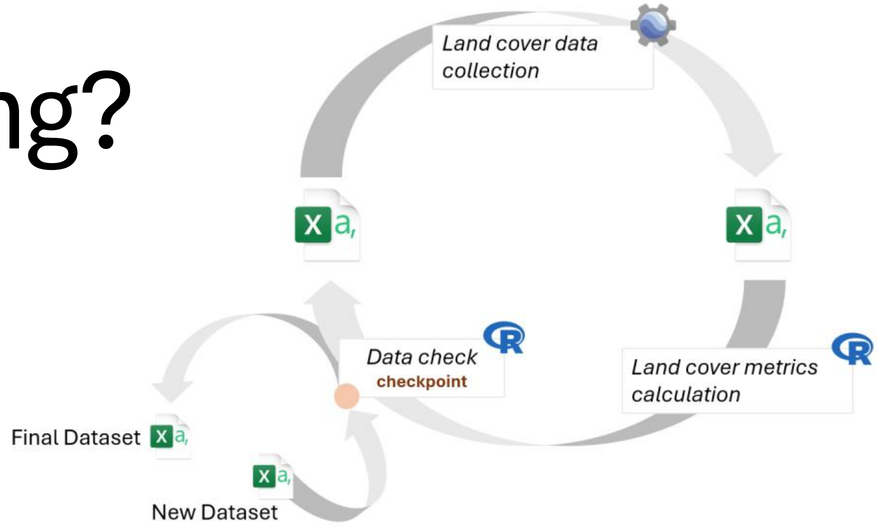
```
lm_metrics_full_selected_calc <-
  lm_metrics_full_selected_calc %>%
  group_by(measurement_id, buffer_radius_m) %>%
  mutate(shannons_index = -(sum(if_else(proportion > 0, proportion * log(proportion), 0))))
```

Calculate **shannons diversity index** as p is the proportion (n/N) of individuals of one particular species found (n) divided by the total number of individuals found (N), ln is the natural log, Σ is the sum of the calculations.

```
data_20241007 <- read.csv("C:\\Users\\lisa7\\Documents\\surrounding_landscapes\\data\\working data\\20

df_20241020 <-
  left_join(data_20241007, lm_metrics_full_selected_calc, join_by("measurement_id"=="measurement_id"))
```

Merge with the original dataset

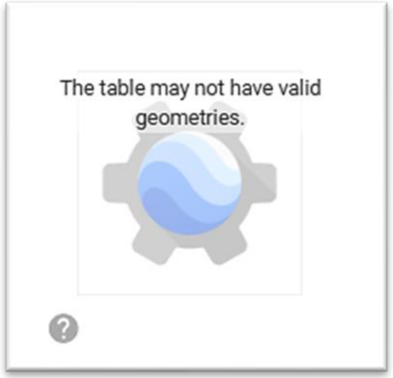
What can go wrong?



- Upload the data with valid coordinates
- Import your data and necessary maps
- Filter your data points by specific year
- Create buffers around your data points
- Export as a csv file

What can go wrong?

```
// (4.4) Process edge length and area for points in a given year
function processMetricsForYear(filteredPoints, image, year) {
  var bufferedPoints = filteredPoints.map(bufferPoints(5000));
  var results = bufferedPoints.map(function(point) {
    var bufferGeom = point.geometry();
    var edges = detectEdges(image, bufferGeom);
```




```
// Iterate over each band (year) in the image
for (var i = 1; i <= 23; i++) {
  var year = 1999 + i; // starts at year 2000 for annual maps
  var layerName = "GLC FCS " + year.toString();
  var band = image.select("b" + i);

// Apply the function to the band and add layer
  addLayer(recodeClasses(band), layerName);
}
```

```
ImageCollection (Error)
  Collection query aborted after accumulating over 5000
  elements.
```

```
FeatureCollection (Error)
  Too many concurrent aggregations.
```

Coordinates are **not valid**

When saving the csv file from R, set row.names to FALSE

Check your data formats and column names

Set your data points to **geometries**

Memory errors

Upload the data with valid coordinates

Import your data and necessary maps

Filter your data points by specific year

Create buffers around your data points

Export as a csv file

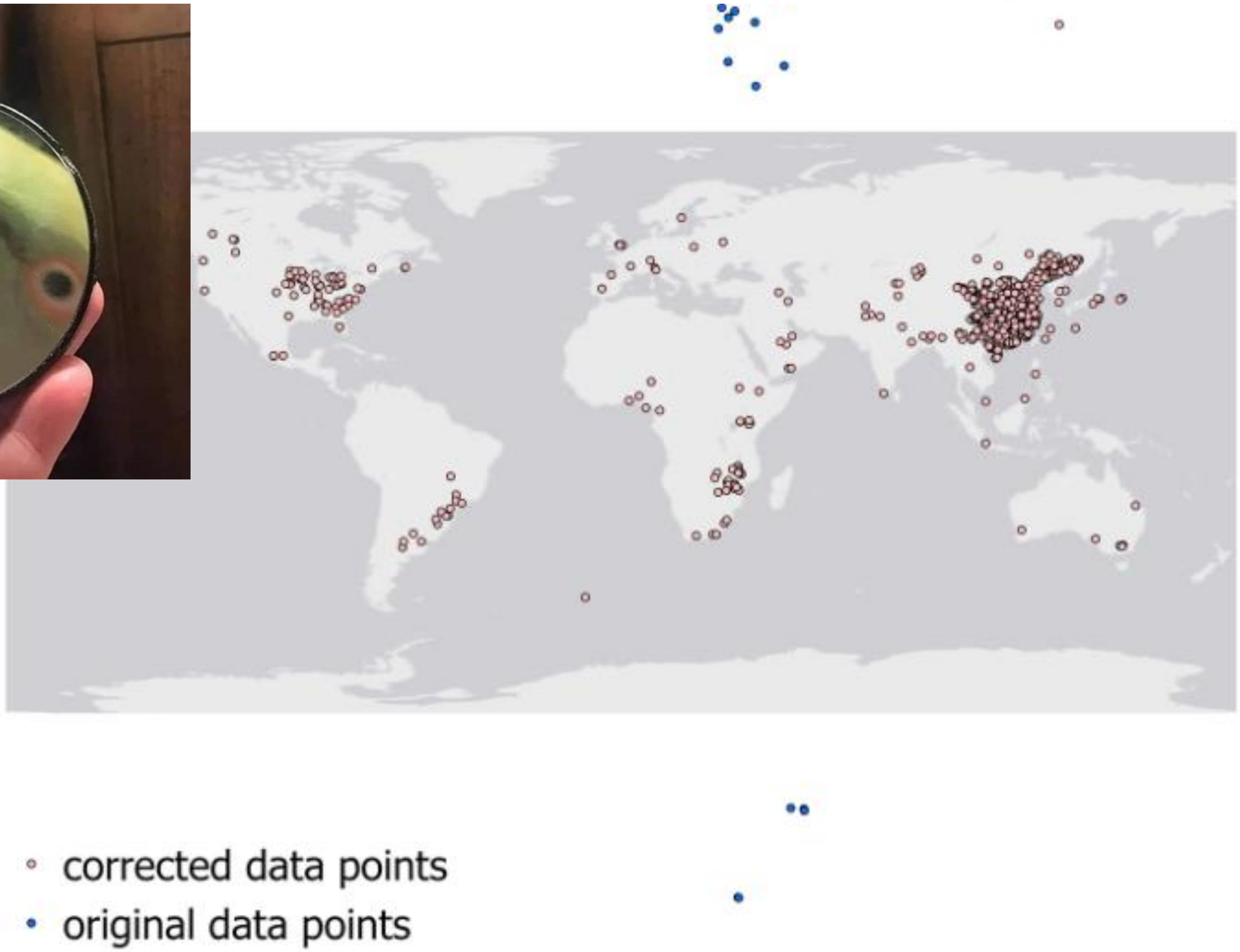
Zero Edge Problem

yield_SD_treatment_kgha	class	area_m2	edgelength_m	buffer_radius_m
NA	200	663.6914	0	2500
18.32993	20	684.7454	0	5000
17.38912	20	684.7454	0	5000
16.14633	20	684.7454	0	5000
10.97894	20	684.7454	0	5000
10.91423	20	684.7454	0	5000
12.23264	20	684.7454	0	5000
18.50342	20	684.7454	0	5000
19.73768	20	684.7454	0	5000
19.85532	20	684.7454	0	5000
18.64996	20	684.7454	0	5000
236.00000	182	739.5312	0	5000
NA	10	661.4300	0	1000
NA	0	83051.0926	0	5000
NA	0	83051.0926	0	5000
NA	0	83051.0926	0	5000
NA	0	83051.0926	0	5000
NA	0	83051.0926	0	5000
NA	0	83051.0926	0	5000

Solution:

- ❑ Check if the map covers all of your observation areas
- ❑ Increase your maximum memory capacity to enable better processing
- ❑ Apply buffers before proceeding to metrics calculation (edge and area)

```
Error:
Set of coordinates activated your trust issues
```



Coordinate cleaner

What can go wrong?

Mixed **coordinates' formats**

How to fix it?

Identify coordinates' formats using regular expressions

Convert all coordinates to decimal degrees

longitude	latitude
116°35′16.24″	39°35′47.03″
116°35′16.24″	39°35′47.03″
116°35′16.24″	39°35′47.03″
116°35′16.24″	39°35′47.03″
116°35′16.24″	39°35′47.03″
116°35′16.24″	39°35′47.03″
116°35′16.24″	39°35′47.03″
116°18′	37°27′
116°18′	37°27′

longitude	latitude
110°42.59′	34°55.51′
110°42.59′	34°55.51′
116°26′	37°9′
116°26′	37°9′
116°26′	37°9′
116°26′	37°9′

longitude	latitude
120°21'22.21"	36°17'57.94"
120°21'22.21"	36°17'57.94"
120°21'22.21"	36°17'57.94"
120°21'22.21"	36°17'57.94"
120°21'22.21"	36°17'57.94"
108°05′	34°18′
108°05′	34°18′
108°4′	34°17′

longitude	latitude
121.670323	42.021619
121.670323	42.021619
121.670323	42.021619
115°42′	37°53′

longitude	latitude
128°05'E	35°08'N
128°05'E	35°08'N
128°05'E	35°08'N
90°50'E	24°75'N
90°50'E	24°75'N
90°50'E	24°75'N
119°3'E	26°1'N
119°3'E	26°1'N

Coordinate cleaner

What can go wrong?

Mixed **coordinates' formats**

Reversed coordinates

Materials and methods

The study, which was carried out for 2 years spanning four planting seasons, was sited in the Teaching and Research two Farm of the Obafemi Awolowo University, Ile-Ife, Nigeria (Lat. 7.29° N and Lat. 4.34° E). The soil was classified at the series level as Oba series and as Ustoxic Dystropepts according to USDA system (Soil Survey Staff 1992). The soil is

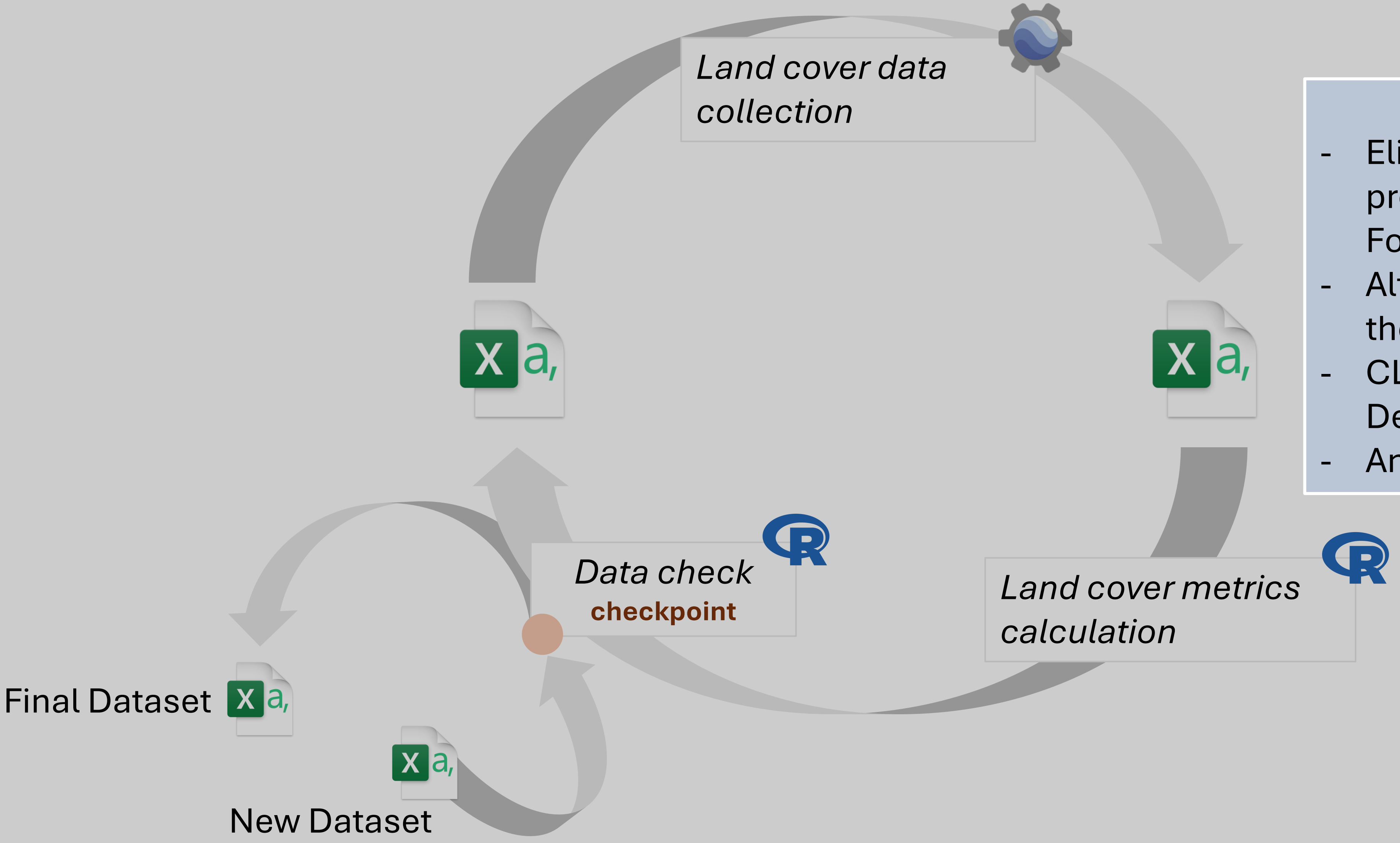
How to fix it?

Identify coordinates' formats using regular expressions

Convert all coordinates to decimal degrees

Look for no-sense coordinate values (|latitude| > 90, E W in latitude etc)

Assign countries based on coordinates and check whether a country could be assigned



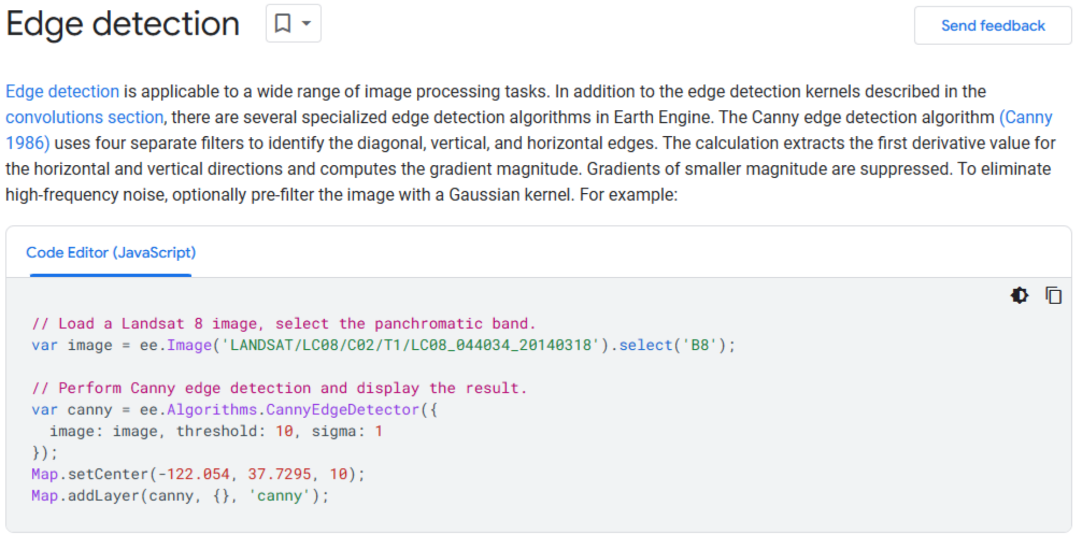
Big thanks to

- Elina Takola for supervising the project and organizing the CLE Forum
- Alfred Burian for his involvement in the data check
- CLE and Remote Sensing Departments
- And all workshop's participants :)

Let's discuss the questions :)

Additional Help and Ressources

- Google Earth Engine Developers
- Google Earth Engine Youtube Channel
- Spatial Thoughts
- Samapriya Roy awesome-gee-community-catalog



Edge detection

Send feedback

Edge detection is applicable to a wide range of image processing tasks. In addition to the edge detection kernels described in the convolutions section, there are several specialized edge detection algorithms in Earth Engine. The Canny edge detection algorithm (Canny 1986) uses four separate filters to identify the diagonal, vertical, and horizontal edges. The calculation extracts the first derivative value for the horizontal and vertical directions and computes the gradient magnitude. Gradients of smaller magnitude are suppressed. To eliminate high-frequency noise, optionally pre-filter the image with a Gaussian kernel. For example:

Code Editor (JavaScript)

```
// Load a Landsat 8 image, select the panchromatic band.
var image = ee.Image('LANDSAT/LC08/C02/T1/LC08_044034_20140318').select('B8');

// Perform Canny edge detection and display the result.
var canny = ee.Algorithms.CannyEdgeDetector({
  image: image, threshold: 10, sigma: 1
});
Map.setCenter(-122.054, 37.7295, 10);
Map.addLayer(canny, {}, 'canny');
```