# Google Earth Engine for hydrology in R

Abdou Khouakhi
*Cranfield University*

@akhouakhi

**Using R In Hydrology**

vEGU 2021

# Outline

- **Introduction to Google Earth Engine (GEE) and `rgee` Package**

- **Quick demo 1: hydrological data retrieval using `rgee`**

- **Quick demo 2: Simple example of machine learning in GEE**

**Notes:**

- This presentation is also accompanied by an R script to execute in Rstudio

- You need to sign up for a Google Earth Engine account to be able to access data and computing capabilities.

# Google Earth Engine

# Google Earth Engine - What is it?

Google Earth Engine is a computing platform that allows users to run geospatial analysis on Google's infrastructure.

The main components of GEE:

- Datasets: A petabyte-scale archive of publicly available remotely sensed imagery and other data.

- Compute power: Google's computational infrastructure optimized for parallel processing of geospatial data.

- APIs: APIs for JavaScript and Python (hosted on GitHub) for making requests to the Earth Engine servers.

- Code Editor: An online Integrated Development Environment (IDE) for rapid prototyping and visualization of complex spatial analyses using the Javascript API.

- Apps: A dynamic, publicly accessible user interface for Earth Engine analyses.

# Data structure in GEE

- Data in GEE are structured in Collections; a stack or time series of images or gridded datasets and features

- For example, all of the images produced by a single sensor are grouped together and presented as a "collection".

- Collections can be filtered, sorted, joined, mapped, or reduced

- An Image can be a Stack of Georeferenced bands

- Each band has its own Mask, Projection, Resolution and a list of properties (Date, bbox..)

Collection

bands
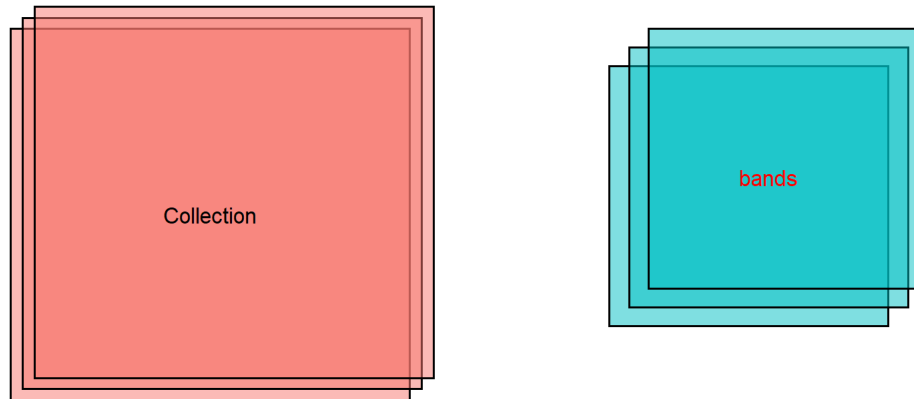
# Interacting with GEE

There are several ways to interact with the platform:

- Explorer

- Javascript Code Editor

- Python wrapper library

- Calling EE from within R

- Using QGIS EE plugin
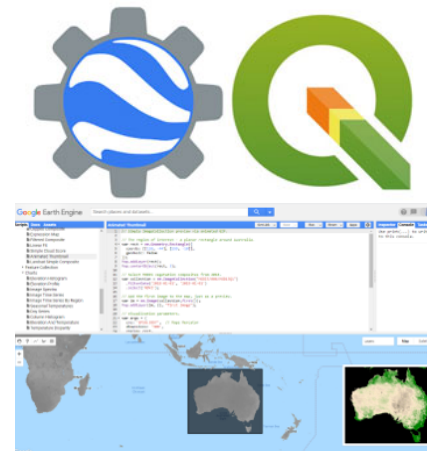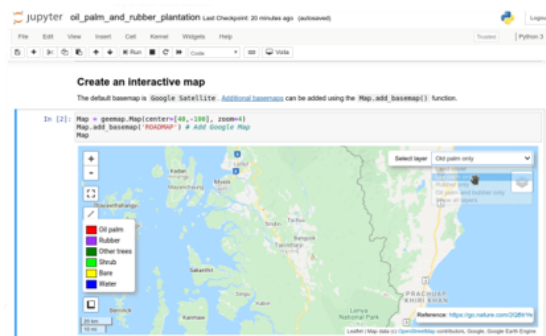
# rgee package (Aybar et al. 2021)

- A package for calling Google Earth Engine API from within R

- Several functions have been implemented to simplify the connection with the R spatial ecosystem

- Details of rgee setup and examples of scripts can be found here

Why rgee?

- Various visualization options
- Many functions to seamlessly interact with GEE API
- Easier transition to a web application such as Shiny
- Reproducibility

# Quick demo: GEE Data retrieval

```
# You will need the following libraries
        library(rgee,quietly = T)
            library(tidyverse)
                library(sf)
            library(lubridate)
             library(stars)
             library(raster)
             library(cptcity)
```

```
# Initialize ee
ee_Initialize(drive = T,quiet = T)
```

# Select data sources

We are going to obtain some hydrological datasets for the upper catchment of the Niger basin. We define an approximate location within the basin, retrieve the basin boundary, river network and other layers.

```r
# get a rough location within a given catchment area
poi <- ee$Geometry$Point(-9.3, 10.38) # change here to your catchment approx. location
```

Available data collections can be found at the GEE data catalog. Alternatively, you can run `ee_utils_dataset_display()`, a helper functions to open a web browser and search the catalog.

```r
# Hydrosheds -Basins - level 4 (change the level for subcatchments)
hybas4 <- "WWF/HydroSHEDS/v1/Basins/hybas_4"
# River network
ffriver <- "WWF/HydroSHEDS/v1/FreeFlowingRivers"
# SMAP soil moisture 10km
smap10km <- "NASA_USDA/HSL/SMAP10KM_soil_moisture"
# Open land map monthly precip 1km
mth_prcp1km <- "OpenLandMap/CLM/CLM_PRECIPITATION_SM2RAIN_M/v01"
# JRC Global Surface Water Mapping Layers 30m
srfce_wtr30m <- "JRC/GSW1_2/GlobalSurfaceWater"
# Copernicus landcover 100m
cgls100m <- "COPERNICUS/Landcover/100m/Proba-V-C3/Global/2019"
# ALOS global digital surface model 30m
alos_dsm <- 'JAXA/ALOS/AW3D30/V3_2'
# Sentinel 2- Surface reflectance
S2SR <- "COPERNICUS/S2_SR"
```

# Retrieve and visualize the data

## Get the data

```r
# get the basin boundary
basin <- ee$FeatureCollection(hybas4)$filterBounds(poi)
# river network
riverNet <- ee$FeatureCollection(ffriver)$filterBounds(basin)
# get July rainfall over the basin
mth_prcp_jul <- ee$Image(mth_prcp1km)$clip(basin)$select('jul')
# get SMAP soil moisture and select ssm band (layer) over the basin
smap_sm <- ee$ImageCollection(smap10km)$filterBounds(basin)$select("ssm")$first()$clip(basin)
# get the surface Water- select frequency with which water was present
jrc_srfce_wtr <- ee$Image(srfce_wtr30m)$select("occurrence")$clip(basin)
# get land cover CGLC 15-19 and select the landcover classes
clc2019 <- ee$Image(cgls100m)$select("discrete_classification")$clip(basin)
# get ALOS global digital surface model 30m and select DSM
alos_elev <- ee$ImageCollection(alos_dsm)$filterBounds(basin)$select("DSM")$mosaic()$clip(basin)
```

## Visualization parameters

```r
# define visualisation parameters
swater_vis <- list(min = 0, max = 100, palette = c('ffffff', 'ffbbbb', '0000ff'))
smap_vis <- list(min = 0.0, max = 28.0, palette =cpt("pj_4_earth" ))
prcp_vis <- list(min = 0, max = 380, palette = cpt("ncl_precip3_16lev"))
elev_vis <- list(min = 0, max = 1500, palette = cpt("gmt_GMT_elevation"))
# initialize a map and add layers
Map$centerObject(basin)
basin_layer <- Map$addLayer(basin,{},"basin")
riverNet_layer <- Map$addLayer(riverNet,list(width = 1.0, color = "blue"),"River network")
jul_prcp_layer <- Map$addLayer(mth_prcp_jul,prcp_vis,"July Precipitation in mm",legend = T)
smapSl_layer <- Map$addLayer(smap_sm,smap_vis,"SMAP soil moisture in mm")
srfce_water_layer <- Map$addLayer(jrc_srfce_wtr,swater_vis,"Surface water occurrence %", legend =
clc_layer <- Map$addLayer(clc2019,{},"Copernicus Land Cover 100m")
elev_layer <- Map$addLayer(alos_elev,elev_vis,"ALOS DMS 30m", legend =T)
```
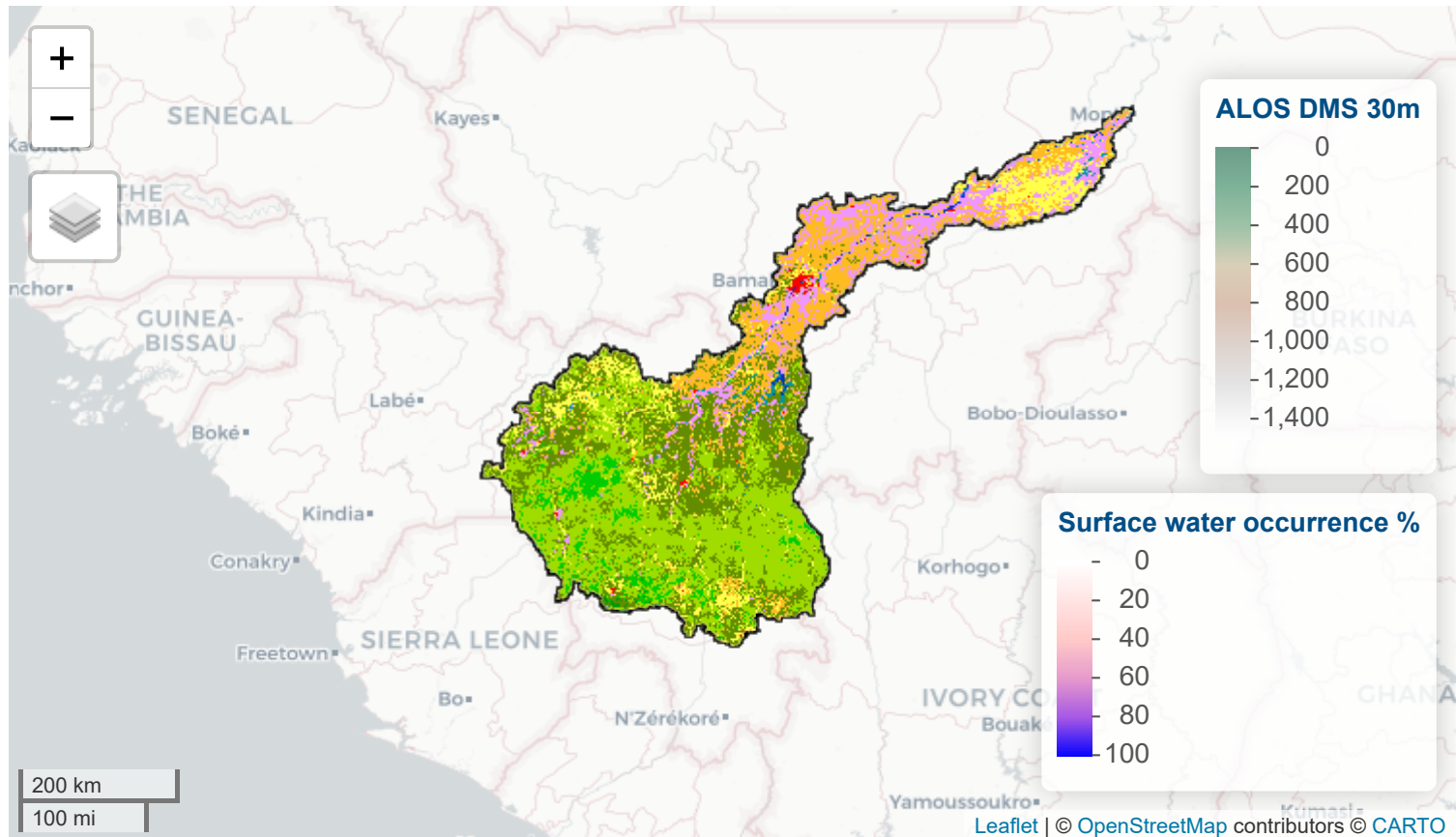
# Add layers to map

```
# show layers
basin_layer+jul_prcp_layer #smapSl_layer|srfce_water_layer
```
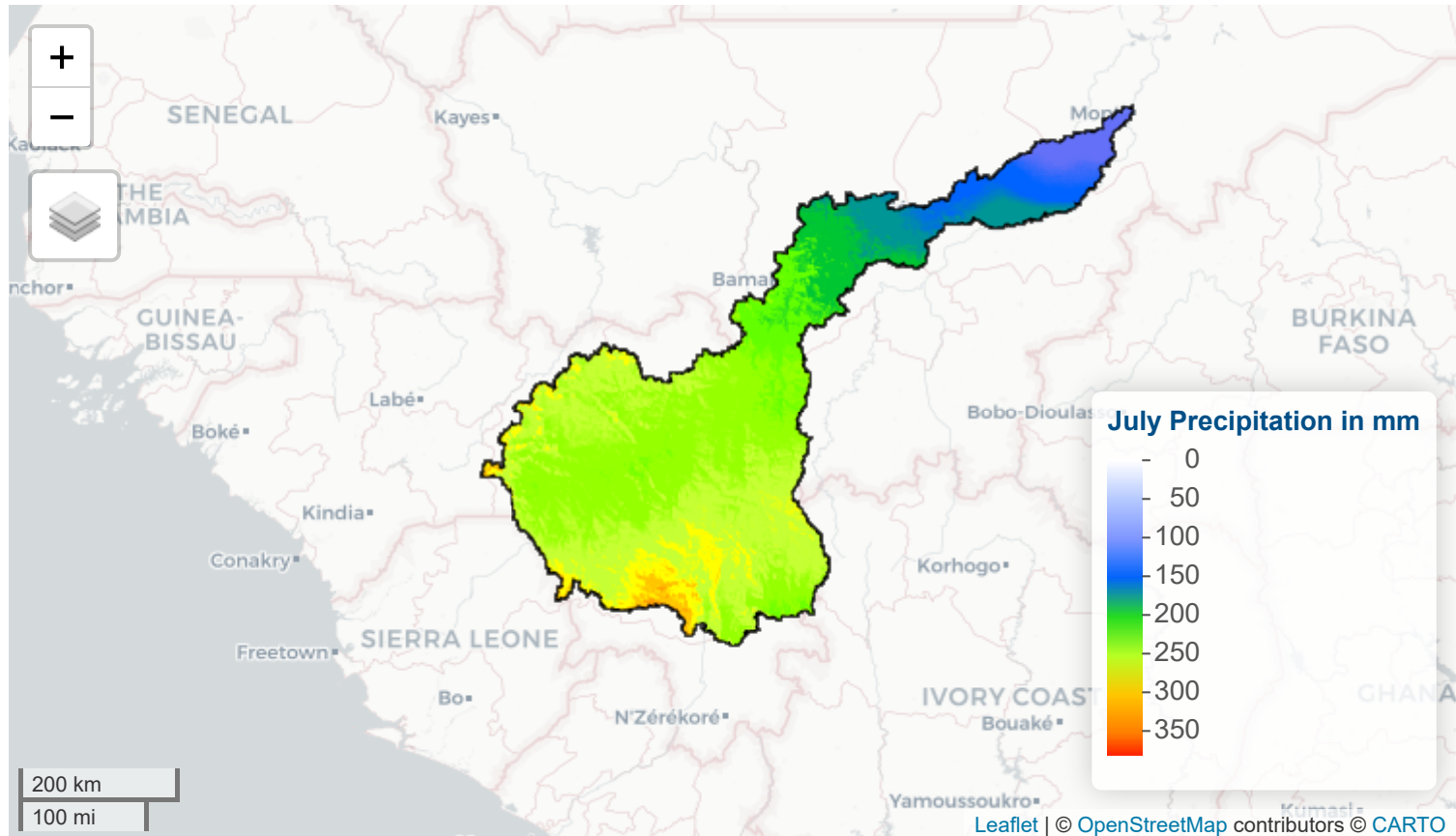
# Add layers to map

```
# add layers
basin_layer+riverNet_layer+srfce_water_layer+smapSl_layer+elev_layer+clc_layer
```

# GEE and ggplot

Here, we use the basin geometry to locally download the gridded rainfall data at the basin level

```r
# basin to ee Geometry
ee_basin <- basin$geometry()
# smap sm to raster
prcp_raster <- ee_as_raster(mth_prcp_jul,
                            region = ee_basin, quiet = T)
# raster to df
ras_todf <- as.data.frame(prcp_raster, xy = T) %>%
  drop_na() %>% filter(jul>0)
# basin to sf
basin_sf <- ee_as_sf(basin)
```

```r
# Integrate GEE data with other ggplot layers
ggplot(basin_sf) +
    geom_raster(
      data = ras_todf,
      aes(x = x, y = y, fill = jul)
    ) +
    scale_fill_gradientn(colours = cpt("ncl_pr
    geom_sf(fill = NA, color = "red") +
    labs(fill = "July precip [mm]")+
    theme_bw()
```
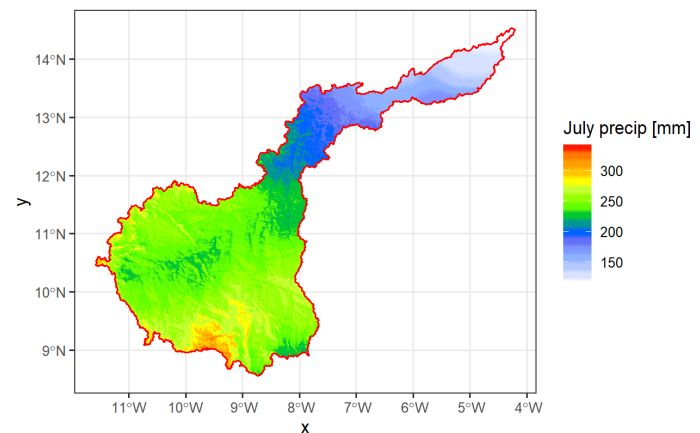
# Quick demo2: identify surface water areas using machine learning within GEE

# Identify water and non-water

- Here, we are going to train a simple machine learning algorithm to distinguish between water and non water areas for a given month at 10 m resolution within our basin using GEE data.
- To train the model, we use Sentinel 2 surface reflectance images, then derive the water and vegetation indices (NDWI and NDVI) in addition to other Sentinel 2 bands.

```r
# get S2 image collection
getS2Image <- function(date, cldPerc) {
  date <- ee$Date(date)
  S2_collection <- (
    ee$ImageCollection("COPERNICUS/S2_SR") # S2_SR collection
    $filterBounds(basin) # filter by basin boundary
    $filterDate(date, date$advance(1, "month")) # filter by month range
    $filter(ee$Filter$lt("CLOUDY_PIXEL_PERCENTAGE", cldPerc)) # filter by cloudy pixel %
  )
  # print metadata
  print(ee_print(S2_collection), max = 1)
  # reduce using median
  S2_median <- S2_collection$
    reduce(ee$Reducer$median())
  # clip to the basin geometry
  S2_median <- S2_median$divide(10000)$clip(basin)
  return(S2_median)
}
```

# Get Sentinel 2SR composite image for a given month

```r
# select a date
dt <-  '2021-01-01'
s2_img <-  getS2Image(date = dt, cldPerc=1) #only cloudless images
```

```
## -------------------------------------------------------------------- Earth Engine ImageCo
## ImageCollection Metadata:
##  - Class                   : ee$ImageCollection
##  - Number of Images        : 189
##  - Number of Properties    : 22
##  - Number of Pixels*       : 6795017397
##  - Approximate size*       : 2.79 TB
## Image Metadata (img_index = 0):
##  - ID                      : COPERNICUS/S2_SR/20210102T104441_20210102T104435_T29PRR
##  - system:time_start       : 2021-01-02 10:57:19
##  - system:time_end         : 2021-01-02 10:57:19
##  - Number of Bands         : 23
##  - Bands names             : B1 B2 B3 B4 B5 B6 B7 B8 B8A B9 B11 B12 AOT WVP SCL TCI_R TCI_G T
##  - Number of Properties    : 81
##  - Number of Pixels*       : 35952473
##  - Approximate size*       : 15.09 GB
## Band Metadata (img_band = 'B1'):
##  - EPSG (SRID)             : WGS 84 / UTM zone 29N (EPSG:32629)
##  - proj4string            : +proj=utm +zone=29 +datum=WGS84 +units=m +no_defs
##  - Geotransform           : 60 0 799980 0 -60 1600020
##  - Nominal scale (meters)  : 60
##  - Dimensions             : 1531 1021
##  - Number of Pixels        : 1563151
##  - Data type              : INT
##  - Approximate size        : 671.97 MB
```

# Identify water and non-water

Calculate NDVI and NDWI and add that to the bands for training

```
# Normalized difference vegetation index
getNDVI <- function (img) {
  img$normalizedDifference(c('B8_median', 'B4_median'))
}
# Normalized difference water index
getNDWI <- function (img) {
  img$normalizedDifference(c('B3_median', 'B8_median'))
}
```

```
# get NDVI and NDWI
S2_NDVI = getNDVI(s2_img)$rename("NDVI")
S2_NDWI = getNDVI(s2_img)$rename("NDWI")
# get info
ee_print(S2_NDVI)
```

```
## ------------------------------------------------------------------------------- Earth Engine Image --
## Image Metadata:
##   - Class                    : ee$Image
##   - ID                       : no_id
##   - Number of Bands          : 1
##   - Bands names              : NDVI
##   - Number of Properties     : 0
##   - Number of Pixels*        : 64800
##   - Approximate size*        : 202.50 KB
## Band Metadata (img_band = NDVI):
##   - EPSG (SRID)              : WGS 84 (EPSG:4326)
##   - proj4string             : +proj=longlat +datum=WGS84 +no_defs
##   - Geotransform            : 1 0 0 0 1 0
##   - Nominal scale (meters)   : 111319.5
##   - Dimensions              : 360 180
##   - Number of Pixels        : 64800
```
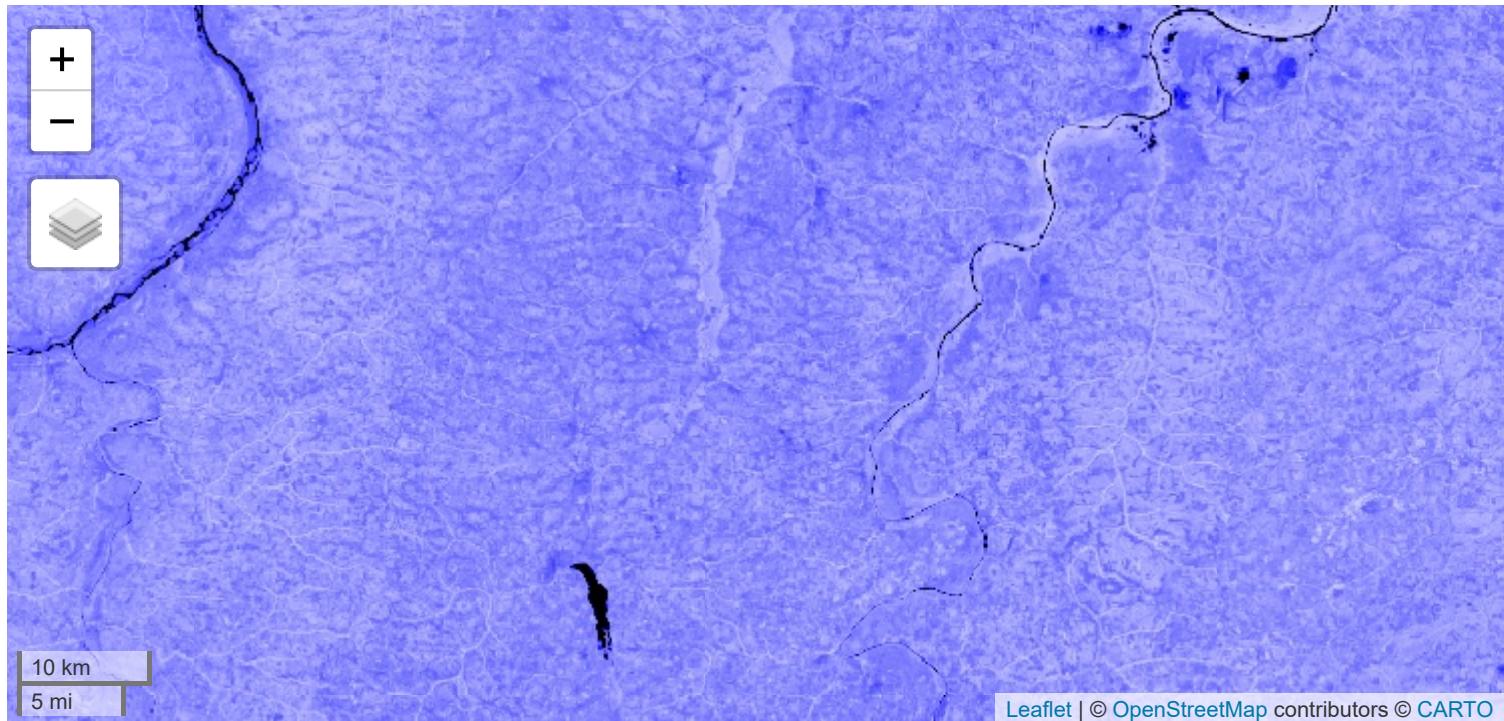
# Identify water and non-water

Stack all the bands together

```
band_stack = (
  ee$Image(s2_img)
  $addBands(S2_NDVI)
  $addBands(S2_NDWI)
  $float()
)
# select only bands with 10 resolution
bands = c( 'NDWI', 'NDVI','B2_median', 'B3_median', 'B4_median', 'B8_median')
band_stack <- band_stack$select(bands)
# get info
ee_print(band_stack)
```

```
## ----------------------------------------------------------------- Earth Engine Image --
## Image Metadata:
##  - Class                  : ee$Image
##  - ID                     : no_id
##  - Number of Bands        : 6
##  - Bands names            : NDWI NDVI B2_median B3_median B4_median B8_median
##  - Number of Properties   : 0
##  - Number of Pixels*      : 388800
##  - Approximate size*      : 1.19 MB
## Band Metadata (img_band = NDWI):
##  - EPSG (SRID)            : WGS 84 (EPSG:4326)
##  - proj4string           : +proj=longlat +datum=WGS84 +no_defs
##  - Geotransform          : 1 0 0 0 1 0
##  - Nominal scale (meters) : 111319.5
##  - Dimensions            : 360 180
##  - Number of Pixels      : 64800
##  - Data type             : FLOAT
##  - Approximate size      : 202.50 KB
## --------------------------------------------------------------------
##  NOTE: (*) Properties calculated considering a constant geotransform and data type.
```

# Visualize the S2 image and the NDWI layers

```
# Display the classification result and the input image.
s2Image_vis = list(bands = c("B4_median", "B3_median", "B2_median"), min = 0, max = 0.3)
class_vis = list(min = 0, max = 1, palette = cpt( "neota_base_blue"))
```

# Create training polygons

Create some training polygons for water and non water areas based on the JRC surface water and S2 image layers

```
# Manually created polygons.
water1 = ee$Geometry$Rectangle(-08.1944, 11.5764, -08.1869, 11.5817)
water2 = ee$Geometry$Rectangle(-08.8728, 10.9167, -08.8714, 10.9174)
nonWater1 = ee$Geometry$Rectangle(-10.6078, 10.0973, -10.6034, 10.1003)
nonWater2 = ee$Geometry$Rectangle(-07.9872, 12.5496, -7.9740, 12.5674)

# Make a FeatureCollection from the above geometries.
tr_polygons = ee$FeatureCollection(c(
  ee$Feature(water1, list(class = 0)),
  ee$Feature(water2, list(class = 0)),
  ee$Feature(nonWater1, list(class = 1)),
  ee$Feature(nonWater2, list(class = 1))
))
```

# Train a Random Forest classifier

Here we sample data using the 4 polygons to train the Random Forest classifier. Please note: there is no model evaluation in this simplified example.

```r
# Get the values for all pixels in each polygon in the training.
training <- band_stack$sampleRegions(
  # Get the sample from the polygons FeatureCollection
  collection = tr_polygons,
  # Keep this list of properties from the polygons.
  properties = list("class"),
  # Set the scale to get S2 image pixels in the polygons
  scale = 10
)
# Create a Random Forest classifier
rf_classifier <- ee$Classifier$smileRandomForest(numberOfTrees = 50)
# Train the classifier.
trained <- rf_classifier$train(training, "class", bands)
# Classify the image.
classified_img <- band_stack$classify(trained)
# Get class #1 of surface water
Water_class <- classified_img$eq(0)
water <- classified_img$updateMask(Water_class)
```

# Visualise the classified surface water

```
# Display the classification result and the input image.
class_vis = list(min = 0, max = 1, palette = c("blue"))
Map$centerObject(basin)
Map$addLayer(s2_img, s2Image_vis," S2_SR image"
) +
Map$addLayer(water,class_vis,"water")
```

# Conclusions

This tutorial:

- Introduces Google Earth Engine and its components
- Introduces the rgee package (Aybar et al. 2021) and how to interact with GEE from within R
- Demonstrates how to retrieve hydrological data from GEE in R
- Demonstrates how to visualize GEE data in ggplot
- Demonstrates how to train a simple Machine Learning model in GEE

Useful links:

- rgee vignette
- rgee examples
- GEE User Guide
- GEE Data Catalogue
- Google Earth Engine Applications (Book)