Google Earth Engine en R

Vol. 01

Antony Barja

Índice

rgee created by : Cesar Aybar, Qiusheng Wu, Lesly Bautista, Roy Yali, Antony Barja

Introducción a rgee

rgee es una "librería cliente" de Earth Engine para R, que permite a los usuarios aprovechar las ventajas que presenta el ecosistema espacial de R dentro de Google Earth Engine y viceversa.
Todas las clases, módulos y funciones de la API de Python de Earth Engine están disponibles en R gracias a la librería reticulate❤️; finalmente rgee adiciona nuevos features como el diseño del imput y output de datos, la visualización en mapas interactivos, la facil extracción de series de tiempo, el manejo y la visualización de metadatos.

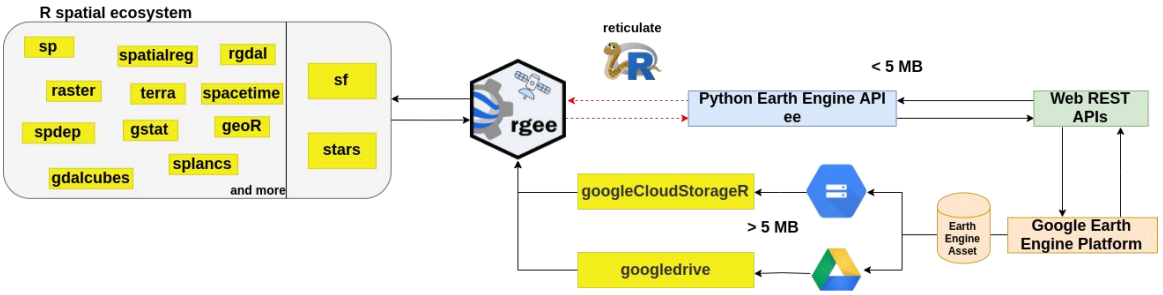


Figura 1: Arquitectura de rgee

Instalación de rgee y otros

Para instalar **rgee** solo necesitamos correr los siguientes comandos:

```
remotes::install_github('r-spatial/rgee')
library(rgee)
ee_install()
ee_Initialize()
```

```
> ee_Initialize()
── rgee 0.6.2 ──────────────────────────────────────────────── earthengine-api 0.1.223 ──
√ email: not_defined
√ Initializing Google Earth Engine:  DONE!
√ Earth Engine user: users/antonybarja8
```

Para poder potencializar nuestro análisis geoespacial vamos a instalar algunas liberías adicionales, estás son las siguientes:

```
install.packages('mapview')   # Para visualizar de forma interáctiva
install.packages('tidyverse') # Para ciencia de datos
install.packages('sf')        # Para manejar datos vectoriales
install.packages('raster')    # Para manejar datos raster
install.packages('cptcity')   # Para manejar paletas de colores
install.packages('ggmap')     # Para manejar tipos de basemap
```

Para activar o llamar cada una de las liberias instaladas, empleamos la siguente función **library() | requiere()**

```
library(mapview)
library(tidyverse)
library(sf)
library(raster)
library(cptcity)
library(ggmap)
```

Sintáxis básica de rgee

rgee presenta una sintaxis muy similar a la de JavaScript o a la de Python como se muestra en la siguiente figura (Fig.02); sin embargo, hay algunas consideraciones que debes de tomar en cuenta, y esto se detalla en el siguiente enlace aqui.

JS (Code Editor)

```
var db = 'CGIAR/SRTM90_V4'
var image = ee.Image(db)
print(image.bandNames())
#> 'elevation'
```

Python

```
import ee
ee.Initialize()
db = 'CGIAR/SRTM90_V4'
image = ee.Image(db)
image.bandNames().getInfo()
#> [u'elevation']
```

R

```
library(rgee)
ee_Initialize()
db <- 'CGIAR/SRTM90_V4'
image <- ee$Image(db)
image$bandNames()$getInfo()
#> [1] "elevation"
```

Figura 2: Sintaxis de GEE en Js, Python y R

Explorando el cátalago de datos de Google Earth Engine

```
ee_search_dataset() %>%
  ee_search_type('ImageCollection') %>%
  ee_search_provider('European Union/ESA/Copernicus') %>%
  ee_search_title('Sentinel-2')
```

id <chr>	provider <chr>
COPERNICUS/S2	European Union/ESA/Copernicus
COPERNICUS/S2_SR	European Union/ESA/Copernicus

2 rows | 1-2 of 11 columns

La función **ee_search_display()** nos permite visualizar el catálogo de imágenes satelitales dentro de la misma plataforma de GEE como se muestra en la siguiente figura (Fig.03)

```
ee_search_dataset() %>%
  ee_search_type('ImageCollection') %>%
  ee_search_provider('European Union/ESA/Copernicus') %>%
  ee_search_title('Sentinel-2') %>%
  ee_search_display()
```

Visualización del cátagalo de Google Earth Engine dentro de R

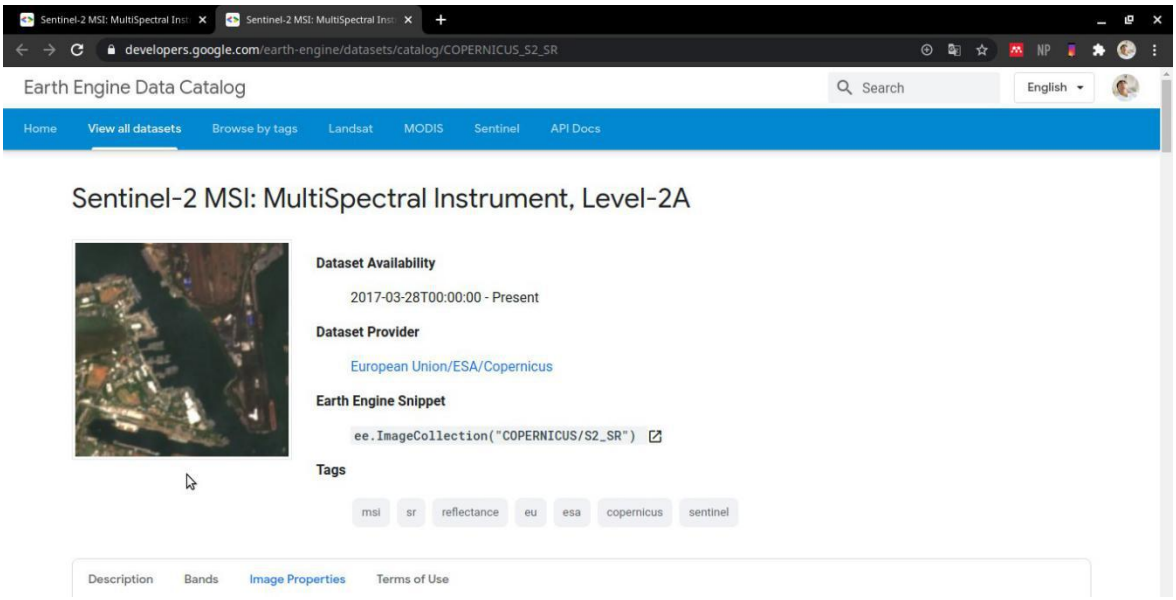


Figura 3: Catálogo de GEE

```
ee_search_dataset() %>%
  colnames()
```

```
[1] "id"         "provider"   "title"      "start_date" "end_date"   "startyear"  "endyear"
[8] "type"       "tags"
```

```
ee_search_dataset() %>%
  select('provider','start_date', 'end_date','type') %>%
  head()
```

	provider <chr>	start_date <chr>	end_date <chr>	type <chr>
1	NOPP	10/01/1992	04/04/2020	ImageCollection
2	EnvirometriX Ltd	01/01/1950	12/31/17	Image
3	CSIRO/SLGA	01/01/1950	12/30/13	ImageCollection
4	European Union/ESA/Copernicus	10/02/2014	04/04/2020	ImageCollection
5	NOAA/NCEP/EMC	06/30/15	04/04/2020	ImageCollection
6	University of Idaho	01/01/1958	11/30/19	ImageCollection

6 rows

Explorando y visualizando imágenes Landsat, Sentinel, MODIS y Aster

Landsat

```
ee_search_dataset() %>%
  select(id) %>%
  filter(str_detect(id, 'LANDSAT')) %>%
  mutate(name = 'LANDSAT') %>%
  head()
```

	id <chr>	name <chr>
1	LANDSAT/LE07/C01/T1_RT	LANDSAT
2	LANDSAT/LO08/C01/T1_RT	LANDSAT
3	LANDSAT/LC08/C01/T1_RT	LANDSAT
4	LANDSAT/LE07/C01/T1_RT_TOA	LANDSAT
5	LANDSAT/LC08/C01/T1_RT_TOA	LANDSAT
6	LANDSAT/LT04/C01/T1_SR	LANDSAT

6 rows

Imágenes de Landsat8 disponibles por fechas para una ubicación especifica:

```
disponible <-
  ee$ImageCollection('LANDSAT/LC08/C01/T1_TOA')$ filterDate(
  '2020-04-01','2020-06-30')$
  filterBounds(ee$Geometry$Point(-71.68,-15.65))

ee_get_date_ic(disponible)
```

id <chr>	time_start <S3: POSIXct>	time_end <S3: POSIXct>
LANDSAT/LC08/C01/T1_TOA/LC08_003071_20200404	2020-04-04 14:47:05	2020-04-04 14:47:05
LANDSAT/LC08/C01/T1_TOA/LC08_003071_20200420	2020-04-20 14:46:59	2020-04-20 14:46:59
LANDSAT/LC08/C01/T1_TOA/LC08_003071_20200506	2020-05-06 14:46:50	2020-05-06 14:46:50
LANDSAT/LC08/C01/T1_TOA/LC08_003071_20200522	2020-05-22 14:46:52	2020-05-22 14:46:52
LANDSAT/LC08/C01/T1_TOA/LC08_003071_20200607	2020-06-07 14:46:59	2020-06-07 14:46:59
LANDSAT/LC08/C01/T1_TOA/LC08_004071_20200411	2020-04-11 14:53:13	2020-04-11 14:53:13
LANDSAT/LC08/C01/T1_TOA/LC08_004071_20200529	2020-05-29 14:53:04	2020-05-29 14:53:04

7 rows

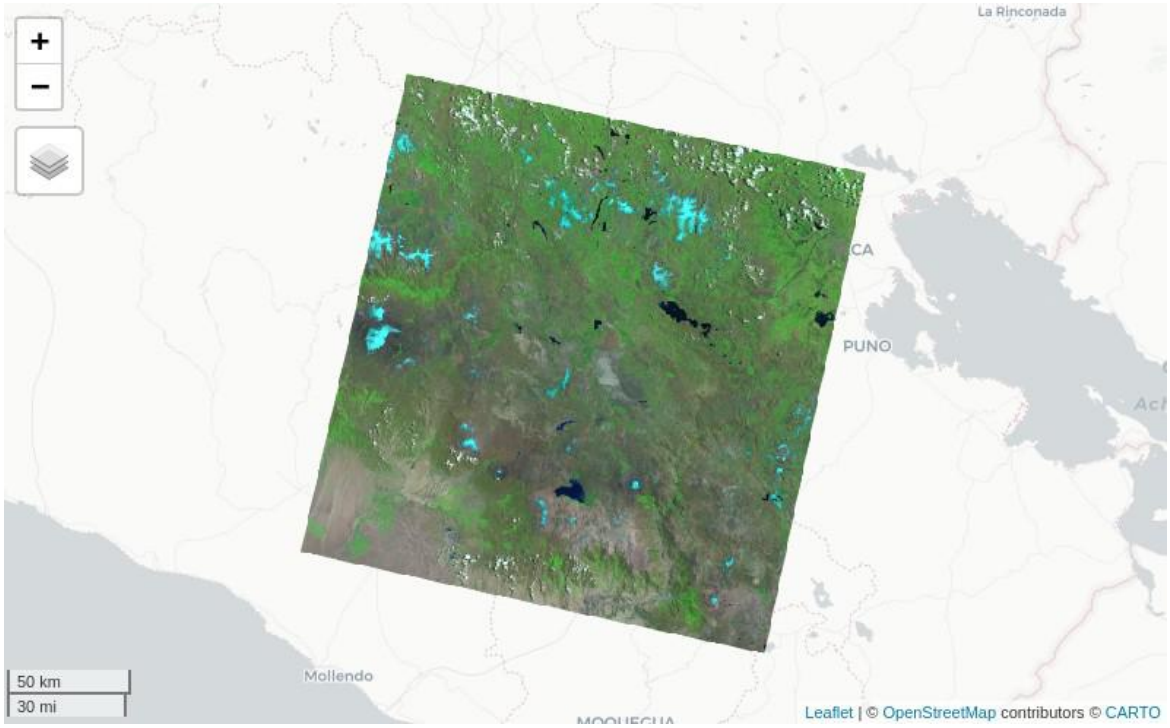
Visualización de la mejor escena:

```
lista <-
  ee$ImageCollection('LANDSAT/LC08/C01/T1_TOA')$ filter
  Date('2020-01-01','2020-07-01')$
  filterBounds(ee$Geometry$Point(-71.68,-
  15.65))$ filterMetadata('CLOUD_COVER','less_than',10)

ee_get_date_ic(lista)
```

id <chr>	time_start <S3: POSIXct>	time_end <S3: POSIXct>
LANDSAT/LC08/C01/T1_TOA/LC08_003071_20200303	2020-03-03 14:47:22	2020-03-03 14:47:22
LANDSAT/LC08/C01/T1_TOA/LC08_003071_20200607	2020-06-07 14:46:59	2020-06-07 14:46:59
LANDSAT/LC08/C01/T1_TOA/LC08_004071_20200207	2020-02-07 14:53:39	2020-02-07 14:53:39
LANDSAT/LC08/C01/T1_TOA/LC08_004071_20200310	2020-03-10 14:53:30	2020-03-10 14:53:30
LANDSAT/LC08/C01/T1_TOA/LC08_004071_20200411	2020-04-11 14:53:13	2020-04-11 14:53:13

5 rows

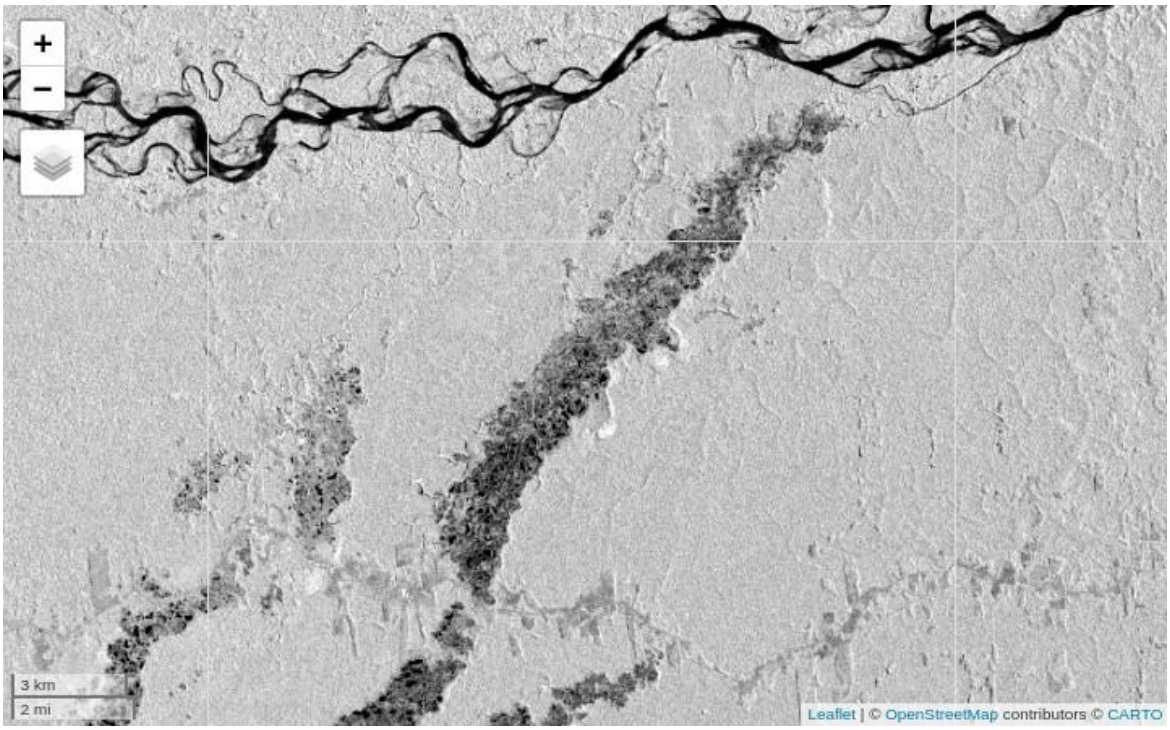
```
viz = list(min = 0,
           max = 0.7,
           bands = c('B7','B5','B4'),
           gamma = 1.75)

landsat <- ee$Image('LANDSAT/LC08/C01/T1_TOA/LC08_003071_20200303')
Map$centerObject(eeObject = landsat,zoom = 8)
Map$addLayer(eeObject = landsat,visParams = viz)
```



Visualización de imágenes sentinel 1 y 2

Sentinel-1

```
latlon <- ee$Geometry$Point(-69.96,-12.84)
coleccionVV <-
  ee$ImageCollection('COPERNICUS/S1_GRD')$ filterDate('
  2016-01-01', '2016-05-31')$
  filter(ee$Filter$eq('instrumentMode',
  'IW'))$ filter(ee$Filter$eq('orbitProperties_pass',
  'ASCENDING'))$ filterMetadata('resolution_meters', 'equals' ,
  10)$ filterBounds(latlon)$
  select('VV')

Map$centerObject(latlon,zoom = 12)
coleccionVV$
  median()%>%
  Map$addLayer(visParams = list(min= -20 , max= -5))
```



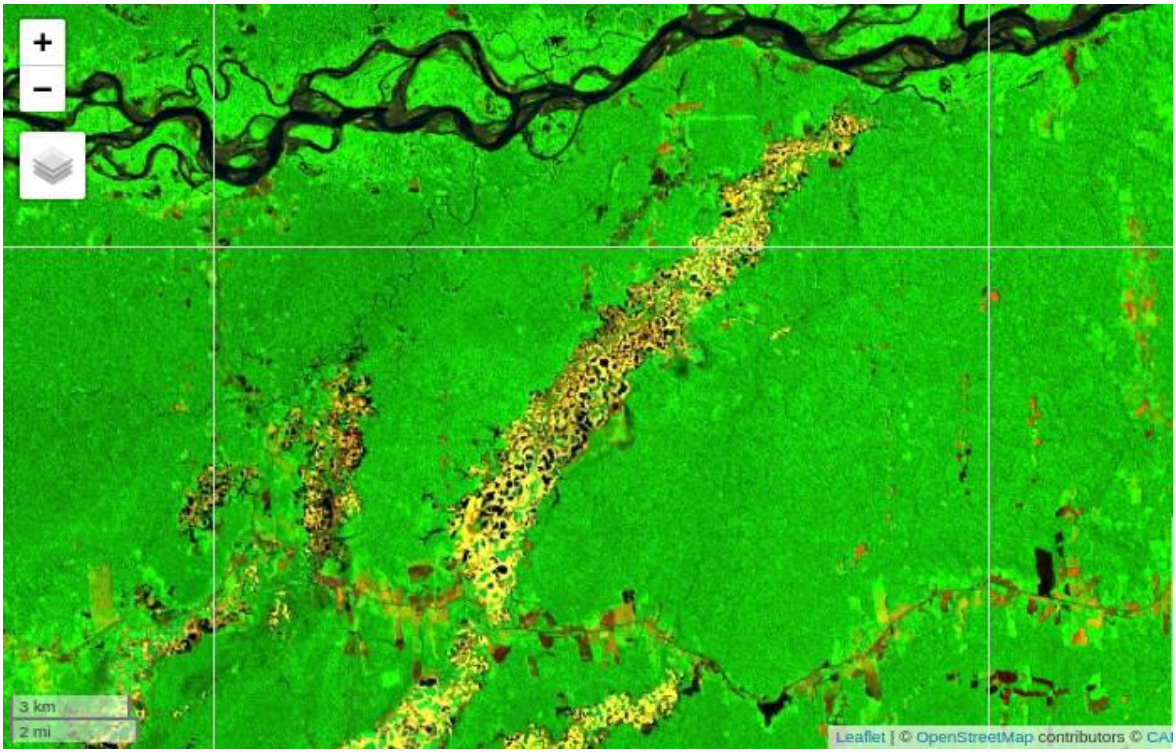
Sentinel-2

```
coleccion_sen2 <- ee$ImageCollection('COPERNICUS/S2')$
filterDate('2016-01-01','2016-12-30')$
filterBounds(latlon)$
filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',5)

ee_get_date_ic(coleccion_sen2)
```

id <chr>	provider <chr>
COPERNICUS/S2	European Union/ESA/Copernicus
COPERNICUS/S2_SR	European Union/ESA/Copernicus

2 rows | 1-2 of 11 columns

```
id <- 'COPERNICUS/S2/20160917T150612_20160917T150614_T19LCF'
sen2 <- ee$Image(id)
Map$centerObject(latlon,zoom = 12)

sen2 %>%
  Map$addLayer(visParams = list(min = 450,
                                max =3500,
                                bands= c('B11','B8A','B2'),
                                gamma = 0.5))
```

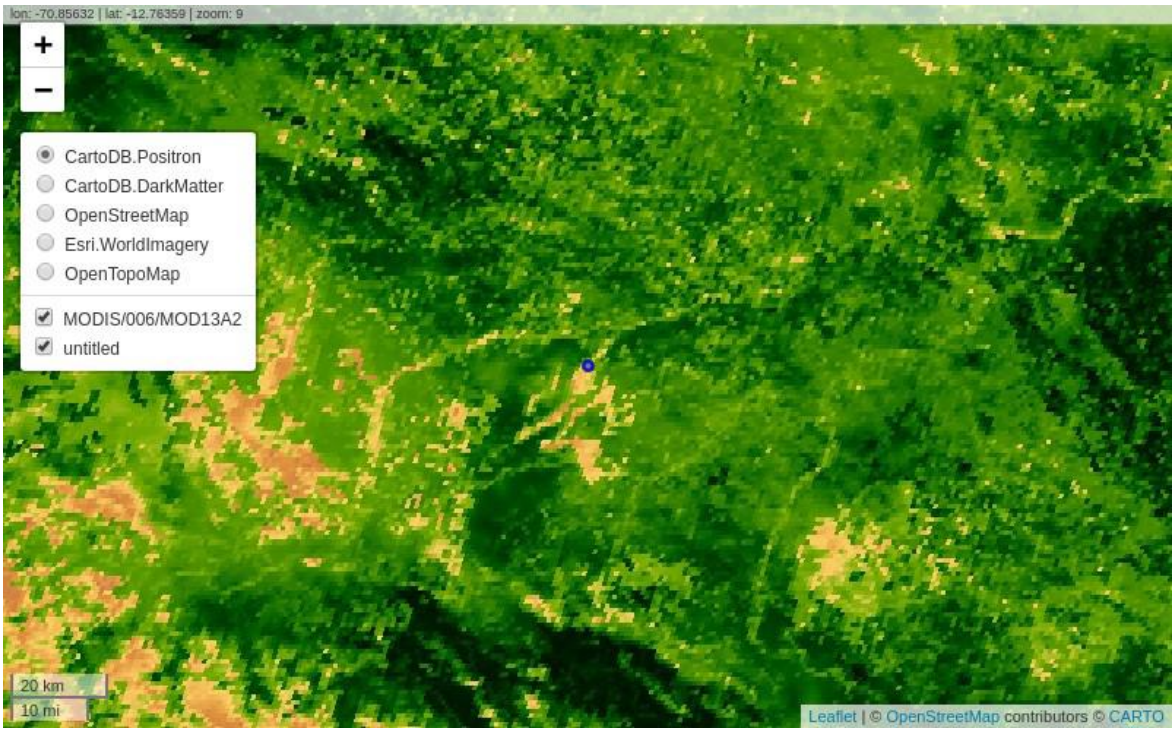


Visualización de imágenes MODIS

```
list_modis <-
  ee$ImageCollection('MODIS/006/MOD13A2')$ filterDate(
  '2016-01-01','2016-12-31')$
  filterBounds(latlon)$
  select("NDVI")
```

Visualizando una escena promedio de un mes específico

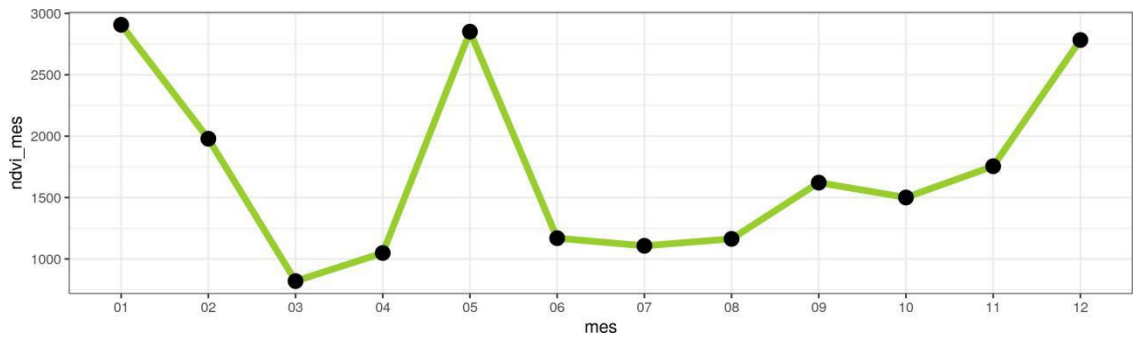
```
modis_feb <- ee$Image(list_modis$filterDate('2016-02-01','2016-02-29')$
mean()
viz <- list(min = 0.0,
              max = 9000.0,
              bands = "NDVI",
              palette = c(
                'FFFFFF', 'CE7E45',
                'DF923D', 'F1B555',
                'FCD163', '99B718',
                '74A901', '66A000',
                '529400', '3E8601',
                '207401', '056201',
                '004C00', '023B01',
                '012E01', '011D01',
                '011301')
            )

Map$centerObject(latlon, zoom = 9)
modis_feb  >
  Map$addLayer(visParams = viz) +
  Map$addLayer(latlon,visParams = list(color = '0518DC'))
```



Con **rgee** puedes analizar series de tiempos de forma rápida y con pocas lineas de código, para está ocación vamos a ver la variación mensual de nuestro punto de control en campo.

obs: **ee_extract()** nos permite estraer los valores pixeles a la geometria asociada

```
ndvi_ts <- ee_extract(list_modis,
                      latlon,
                      fun = ee$Reducer$mean())

colnames(ndvi_ts) <- sprintf("%02d", 1:12)

ndvi_ts %>%
  reshape2::melt() %>%
  separate(variable,into = c("año","mes","día"),sep = "_") %>%
  group_by(mes) %>%
  summarise(ndvi_mes =mean(value)) %>%
  mutate(id = 1) %>%
  ggplot(aes(x = mes,y = ndvi_mes)) +
  geom_line(aes(group = id),color = "#9ACD32",lwd = 2) +
  geom_point(size = 4) +
  theme_bw()
```



Visualización de imágenes ASTER

```
list_aster <-
  ee$ImageCollection('ASTER/AST_L1T_003')$ filterDate(
  '2016-01-01', '2018-12-15')$
  filterBounds(latlon)$ filterMetadata('CLOUDCOVER','less_
  than',1)
```

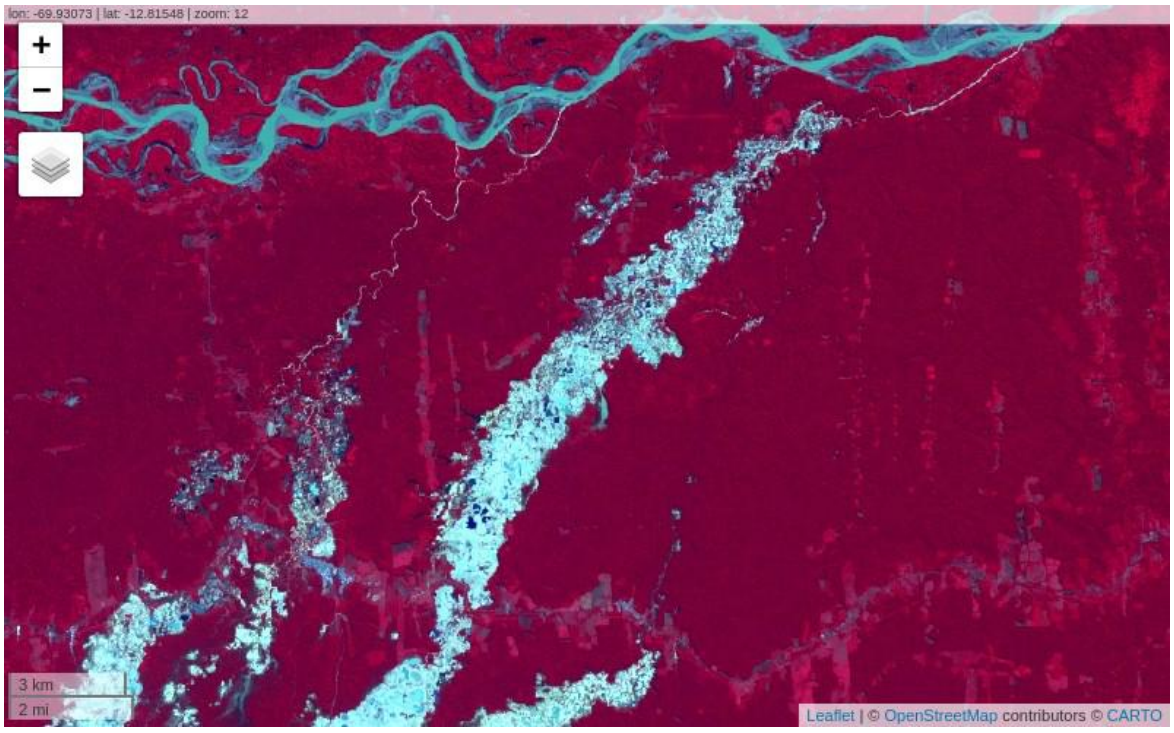
```
ee_get_date_ic(list_aster)
```

id <chr>	time_start <S3: POSIXct>
ASTER/AST_L1T_003/20170616145800	2017-06-16 14:58:00
ASTER/AST_L1T_003/20180728150533	2018-07-28 15:05:33

2 rows

Seleccionamos la segunda escena

```
id <- 'ASTER/AST_L1T_003/20180728150533'
Map$centerObject(latlon,zoom = 12)

ee$Image(id) %>%
  Map$addLayer(visParams = list(min = 25,
                                max = 150,
                                bands = c('B3N','B02','B01'),
                                gamma = 1.2))
```

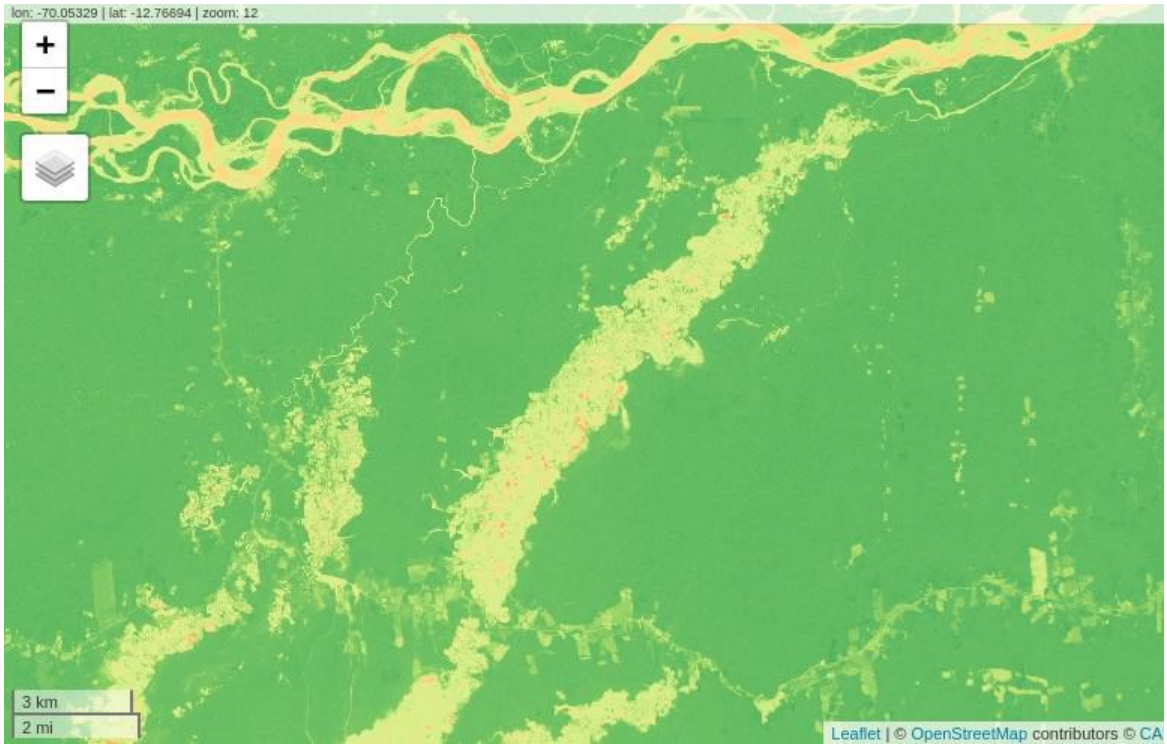


Cálculo de índices espectrales

Dentro de R puedes crear tus propias funciones y puedes calcular cualquier índices espectral, pero existen algunas funciones nativas dentro de rgee como **normalizedDifference** que te permiten calcular el ndvi y otros índices derivados.

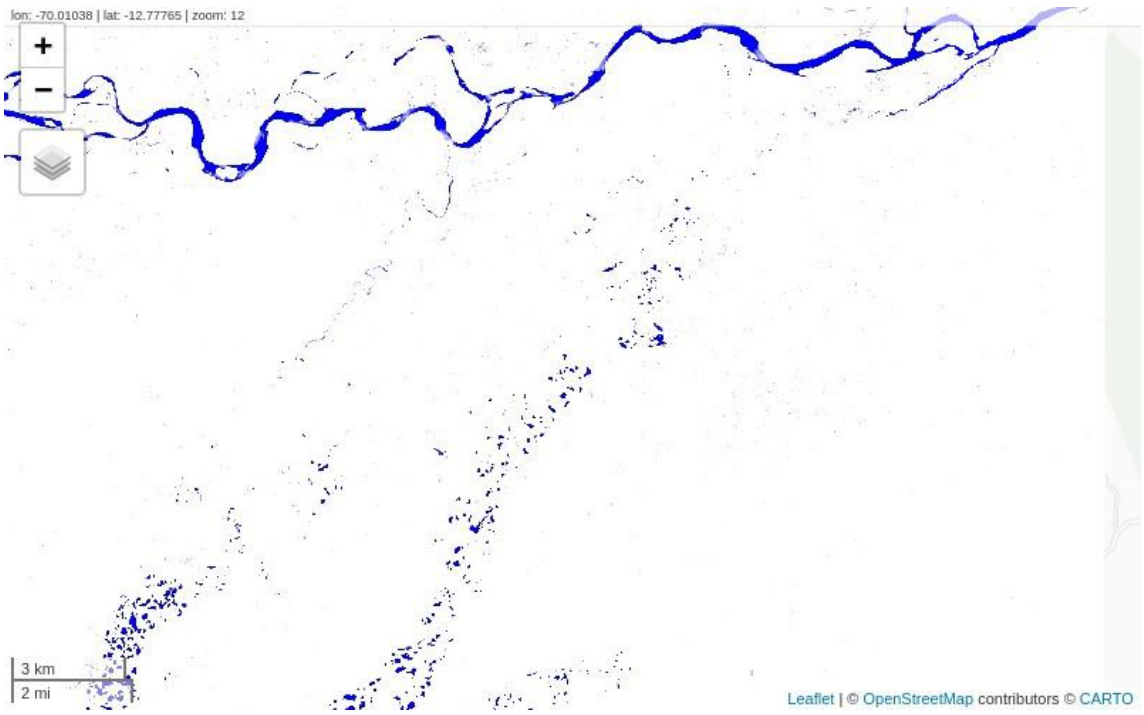
NDVI en Sentinel2

```
Map$centerObject(latlon,zoom = 12)
viz <- list(palette = c(
  "#d73027", "#f46d43",
  "#fdae61", "#fee08b",
  "#d9ef8b", "#a6d96a",
  "#66bd63", "#1a9850")
  )

sen2$normalizedDifference(c('B8A','B4')) %>%
  Map$addLayer(visParams = viz)
```



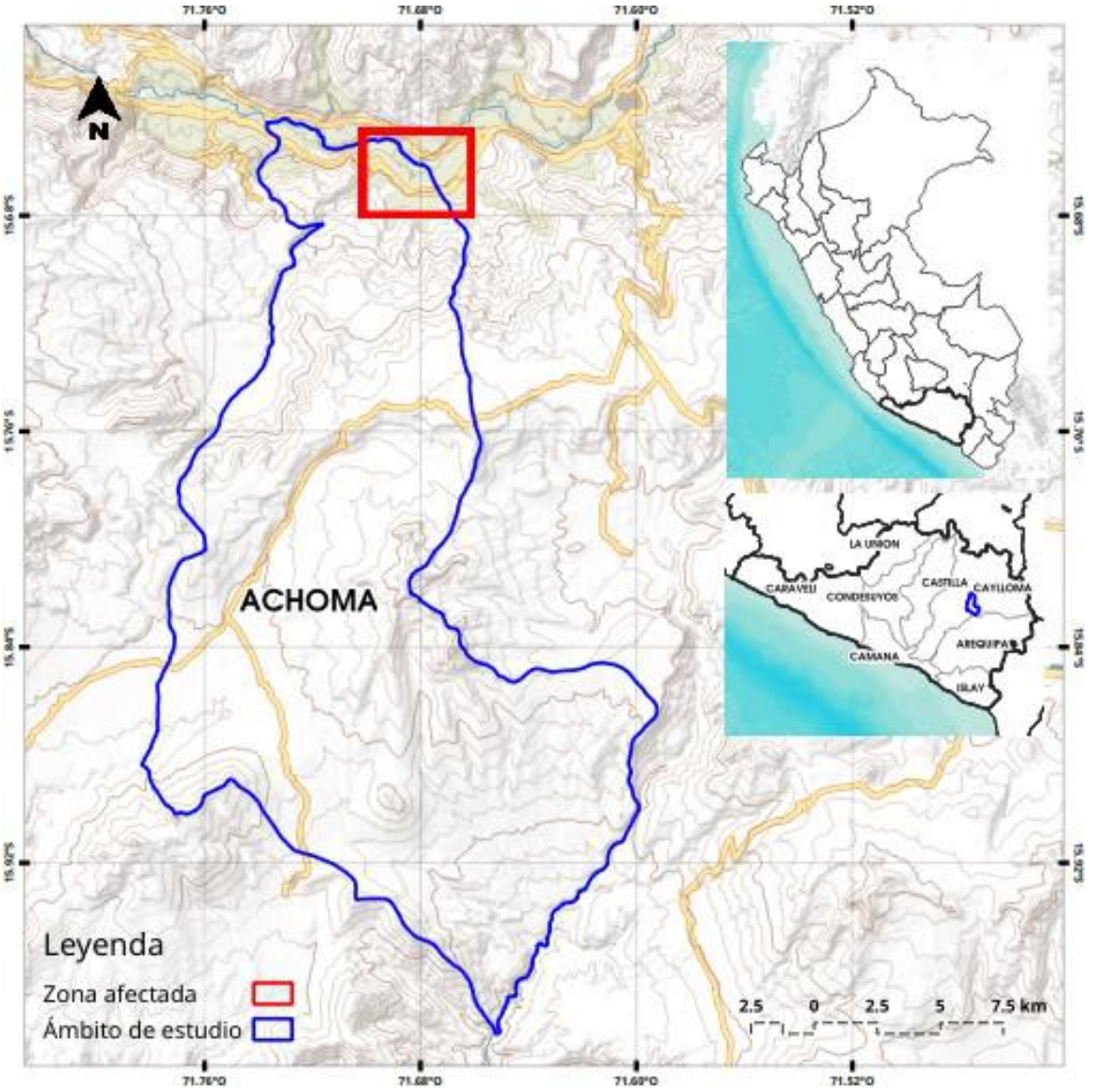
NDWI en Sentinel2

```
viz <-
  list( min =
  -0.15,
  max = 0.65,
  palette = c(
    '#ffff', '#ffff', '#ffff',
    '#ffff', '#ffff', '#0000ff',
    '#0000ff')
  )

sen2$normalizedDifference(c('B8','B11')) %>%
  Map$addLayer(visParams = viz)
```



Caso prático:Mapeo de deslizamientos con rgee

Ámbito de estudio



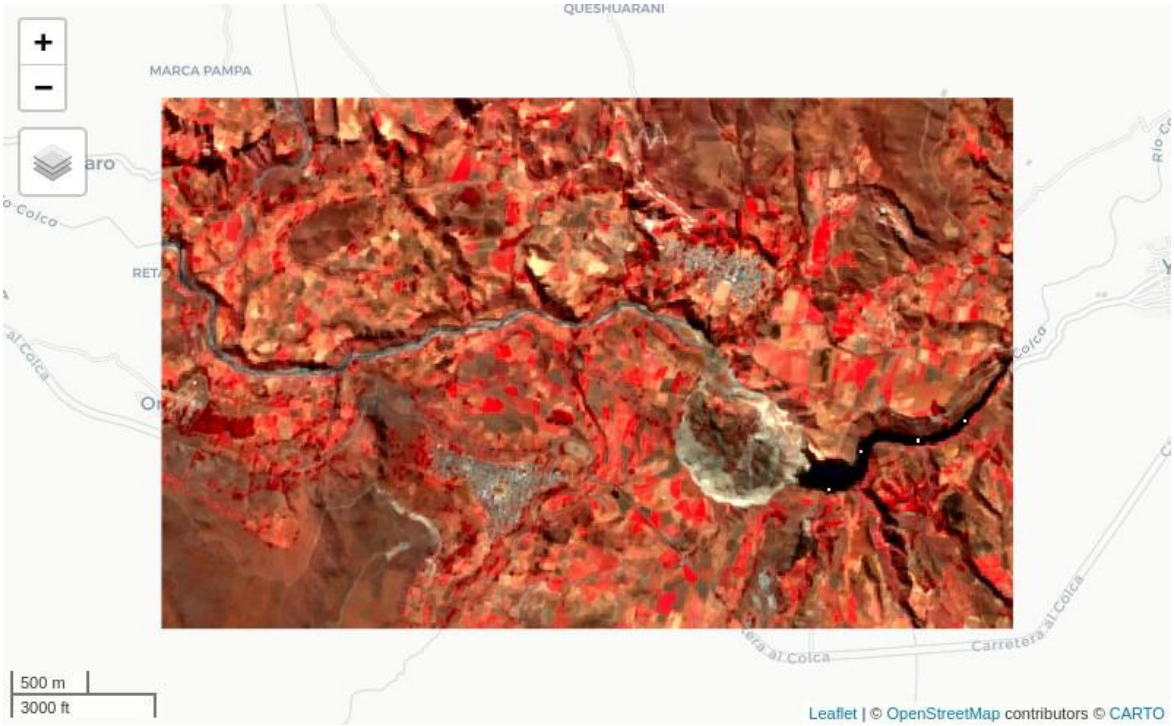
Obtención de imágenes Sentinel - 2

```
box <- ee$Geometry$Rectangle(coords = c(-71.72,-15.67,-71.67,-15.64),
                             proj = "EPSG:4326",
                             geodesic = FALSE)
coleccion_sen2 <- ee$ImageCollection('COPERNICUS/S2_SR')$
  filterDate('2020-06-23','2020-06-28')$
  filterBounds(box)$ filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than'
  ,40)

ee_get_date_ic(coleccion_sen2)
```

id
<chr>
COPERNICUS/S2_SR/20200623T145729_20200623T150542_T18LZH

1 row | 1-1 of 2 columns

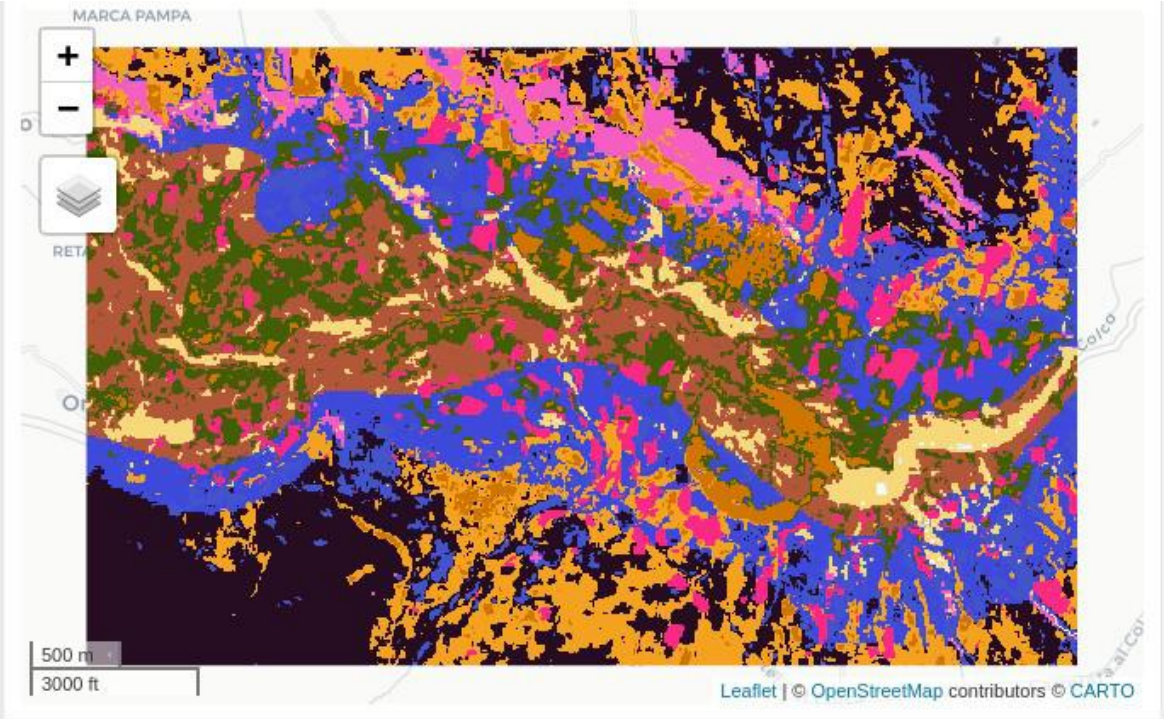
```
id_img <- 'COPERNICUS/S2_SR/20200623T145729_20200623T150542_T18LZH'
sen2 <-
  ee$Image(id_img)$ clip(bo
  x)
Map$centerObject(box)

viz <-list(min = 450,
           max =3500,
           bands= c('B8A','B4','B3'),
           gamma = 1.2)

Map$addLayer(sen2,visParams = viz)
```



Clasificación No Supervisada

```
# Conjunto de datos de entrenamiento
training <-
  sen2$sample( region = box,
  scale = 10,
  numPixels = 10000
)

# Generar el número cluster y entrenear
clusterer <- ee$Clusterer$wekaKMeans(10)$train(training)
```

```
# Clasificar la imágenes usando el cluster entrenado
result <- sen2$cluster(clusterer)

# Visualización de la imágen clasificada
Map$centerObject(box)
Map$addLayer(
  eeObject = result$randomVisualizer(),
  name = "clusters"
)
```



Transformar un objeto ee.Image() a un objeto Raster*

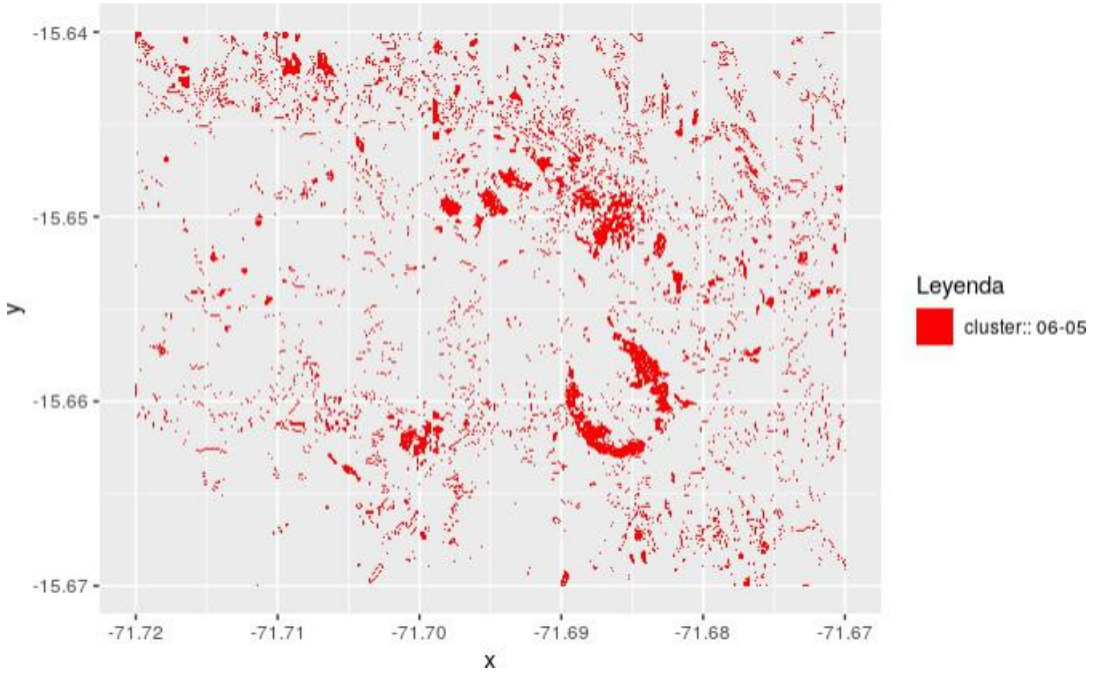
rgee tiene funciones para transformar un objeto de EarthEngine a un objeto de tipo raster* dentro de R, para esto vamos usar la función **ee_as_raster()** como se muestra en la siguiente linea de código:

```
result %>%
  ee_as_raster(region = box,
               scale = 10,) -> img_clas
```

```
- region parameters
 WKT      : POLYGON ((851667.7 8264846, 857033.7 8264762, 857085.9 8268085,
851719.1 8268169, 851667.7 8264846))
 CRS      : 32718
 geodesic : FALSE
 evenOdd  : TRUE
```

Seleccionamos el mejor cluster que carácteriza la amenaza (cluster 6 y 5)

```
img_clas %>%
  projectRaster(crs = '+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs ') %>%
  as('SpatialPixelsDataFrame') %>%
  subset(cluster <= 6 & cluster > 5 ) %>%
  raster() %>%
  as.data.frame(xy = TRUE) %>%
  na.omit() %>%
  mutate(id = 1) -> clase_des

# Visualización simple usando ggplot
clase_des %>%
  ggplot() +
  geom_tile(aes(x = x, y = y, fill = factor(id,labels = 'cluster:: 06-05')))+
    scale_fill_manual(values = "red",name= "Leyenda")
```



Elaboración de mapa final

```
us <- c(left = -71.72,
        bottom = -15.67,
        right = -71.67,
        top = -15.64)

get_map(us,
        zoom = 14,
        maptype = "satellite") %>%
  ggmap() +
```

```
  geom_tile(data = clase_des,
            aes(x = x,
                y = y,
                fill = factor(id,
                              labels = "cluster_06_05"))) +
  scale_fill_manual(values = "red",
                    name= "Map") +
  ggtitle("Kmeans-clustering - Sentinel2") +
  theme(plot.title = element_text(lineheight =.4,
                                  face="bold"),
        legend.position = c(0.10,0.05))
```