Google Earth Engine en R

Table of Contents

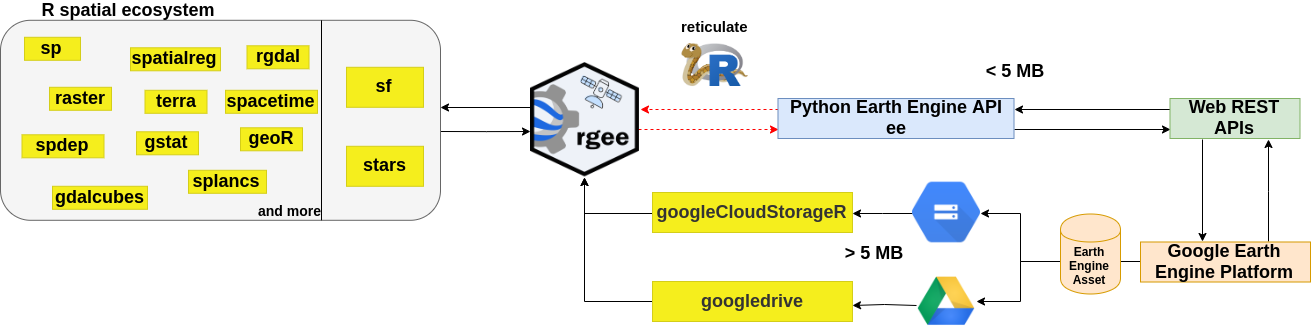
### 

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 <3 ; 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.

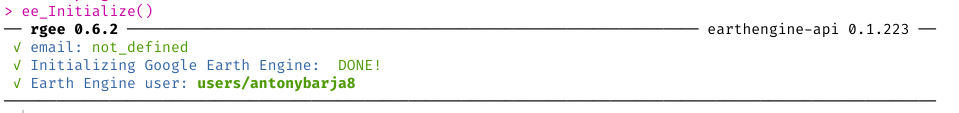


Arquitectura de rgee

## 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()



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

install.packages('mapview') # Pkgs visualizar de forma interáctiva  
install.packages('tidyverse') # Pkgs para ciencia de datos   
install.packages('sf') # Pkgs para manejar datos vectoriales  
install.packages('stars') # Pkgs para manejar datos raster  
install.packages('cptcity') # Pkgs para manejar paletas de colores  
install.packages('tmap') # Pkgs para elaborar mapas temáticos

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

library(mapview)  
library(tidyverse)  
library(sf)  
library(stars)  
library(cptcity)  
library(tmap)

## Sintáxis básica de rgee

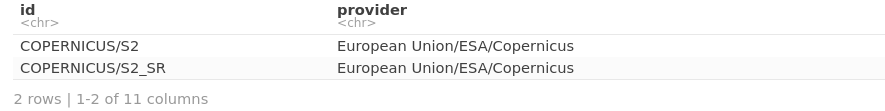
rgee presenta una sintaxis muy similar a la de JavaScript o a la de Python como se muestra en la siguiente figura (Fig.02); sin embargo, hay algunas consideraciones que debes de tomar en cuenta, y esto se detalla en el siguiente enlace [**aquí.**](https://r-spatial.github.io/rgee/articles/considerations.html)



Sintaxis de GEE en Js, Python y R

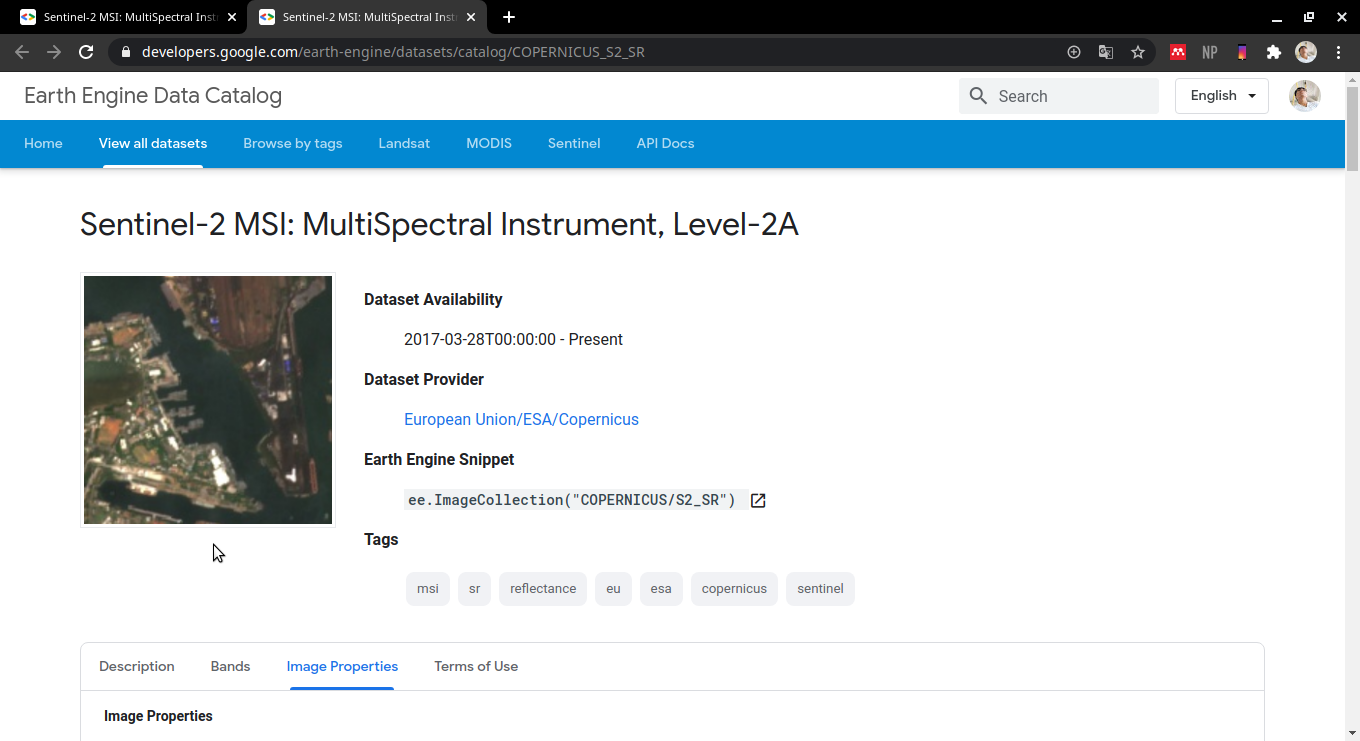
## Explorando el cátalago de datos de Google Earth Engine

ee\_search\_dataset() %>%   
 ee\_search\_type('ImageCollection') %>%   
 ee\_search\_provider('European Union/ESA/Copernicus') %>%   
 ee\_search\_title('Sentinel-2')



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 fig

ee\_search\_dataset() %>%   
 ee\_search\_type('ImageCollection') %>%   
 ee\_search\_provider('European Union/ESA/Copernicus') %>%  
 ee\_search\_title('Sentinel-2') %>%   
 ee\_search\_display()



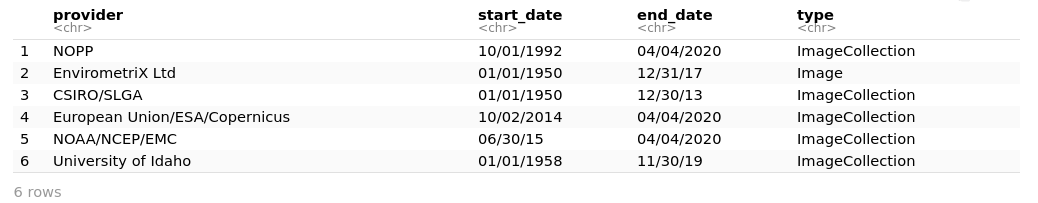
Catálogo de GEE

Visualización del cátagalo de Google Earth Engine dentro de R

ee\_search\_dataset() %>%   
 colnames()

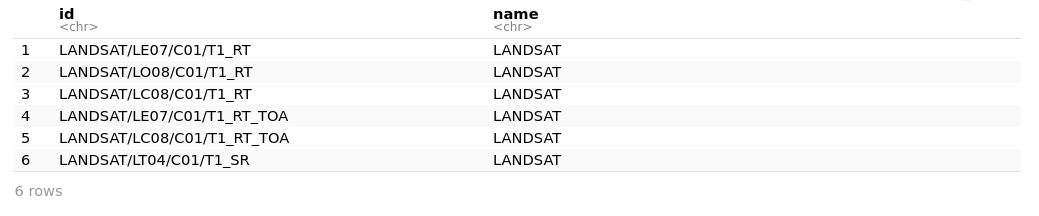


ee\_search\_dataset() %>%   
 select('provider','start\_date', 'end\_date','type') %>%   
 head()



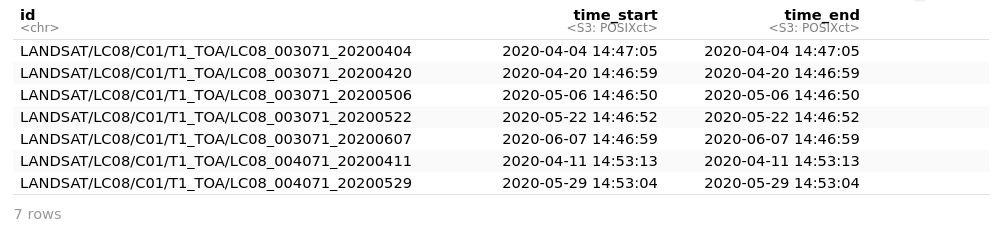
## Explorando y visualizando imágenes Landsat,Sentinel,MODIS y Aster

ee\_search\_dataset() %>%   
 select(id) %>%   
 filter(str\_detect(id, 'LANDSAT')) %>%   
 mutate(name = 'LANDSAT') %>%   
 head()



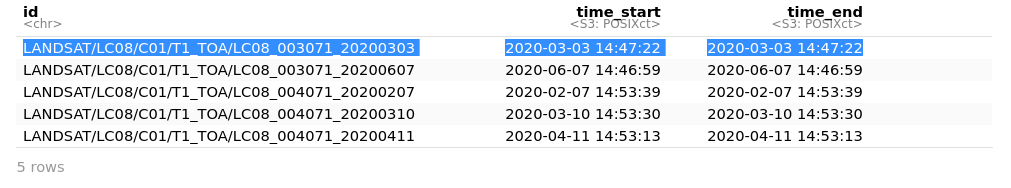
Imágenes de Landsat8 disponibles por fechas para una ubicación especifica:

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)

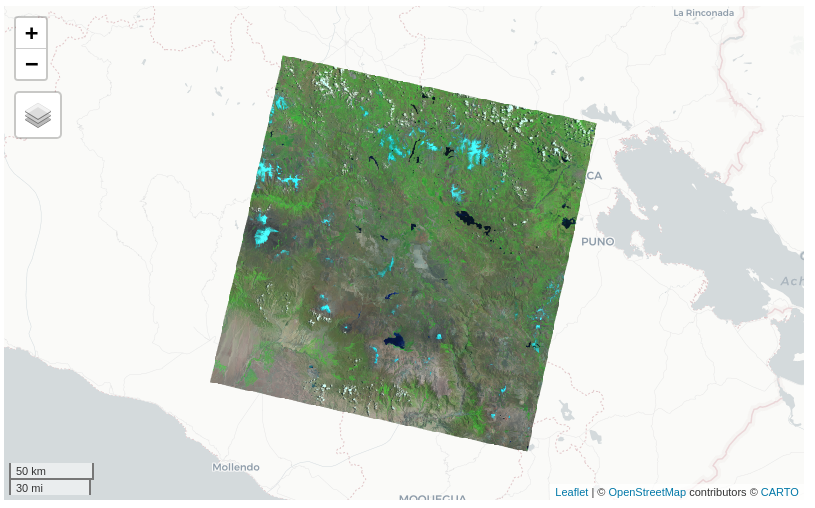


Visualización de la mejor escena:

lista <- ee$ImageCollection('LANDSAT/LC08/C01/T1\_TOA')$  
 filterDate('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)



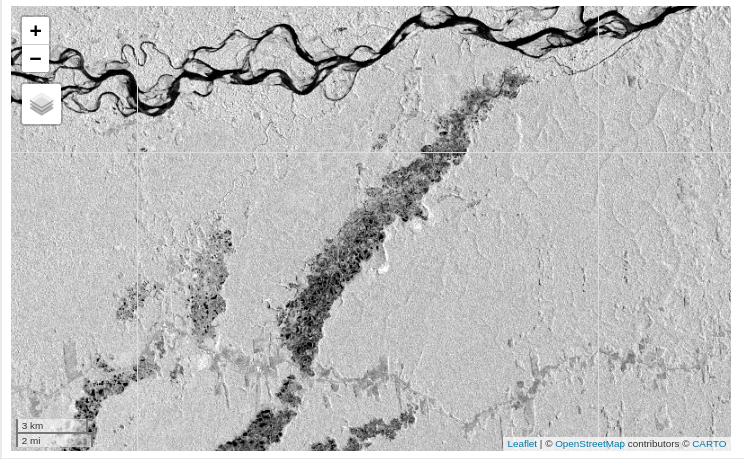
viz = list(min = 0,  
 max = 0.7,  
 bands = c('B7','B5','B4'),  
 gamma = 1.75)  
  
landsat <- ee$Image('LANDSAT/LC08/C01/T1\_TOA/LC08\_003071\_20200303')  
Map$centerObject(eeObject = landsat,zoom = 8)  
Map$addLayer(eeObject = landsat,visParams = viz)



Visualización de imágenes sentinel 1, 2, 5

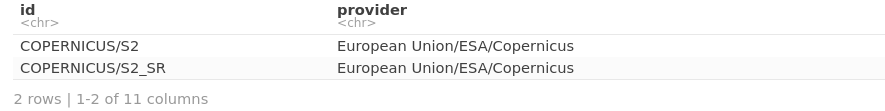
**Sentinel-1**

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 <- '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))

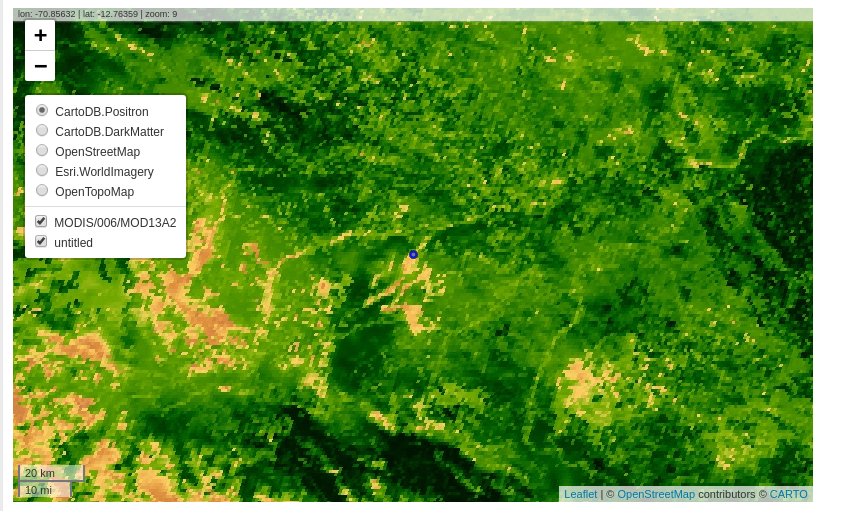


Visualización de imágenes MODIS

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

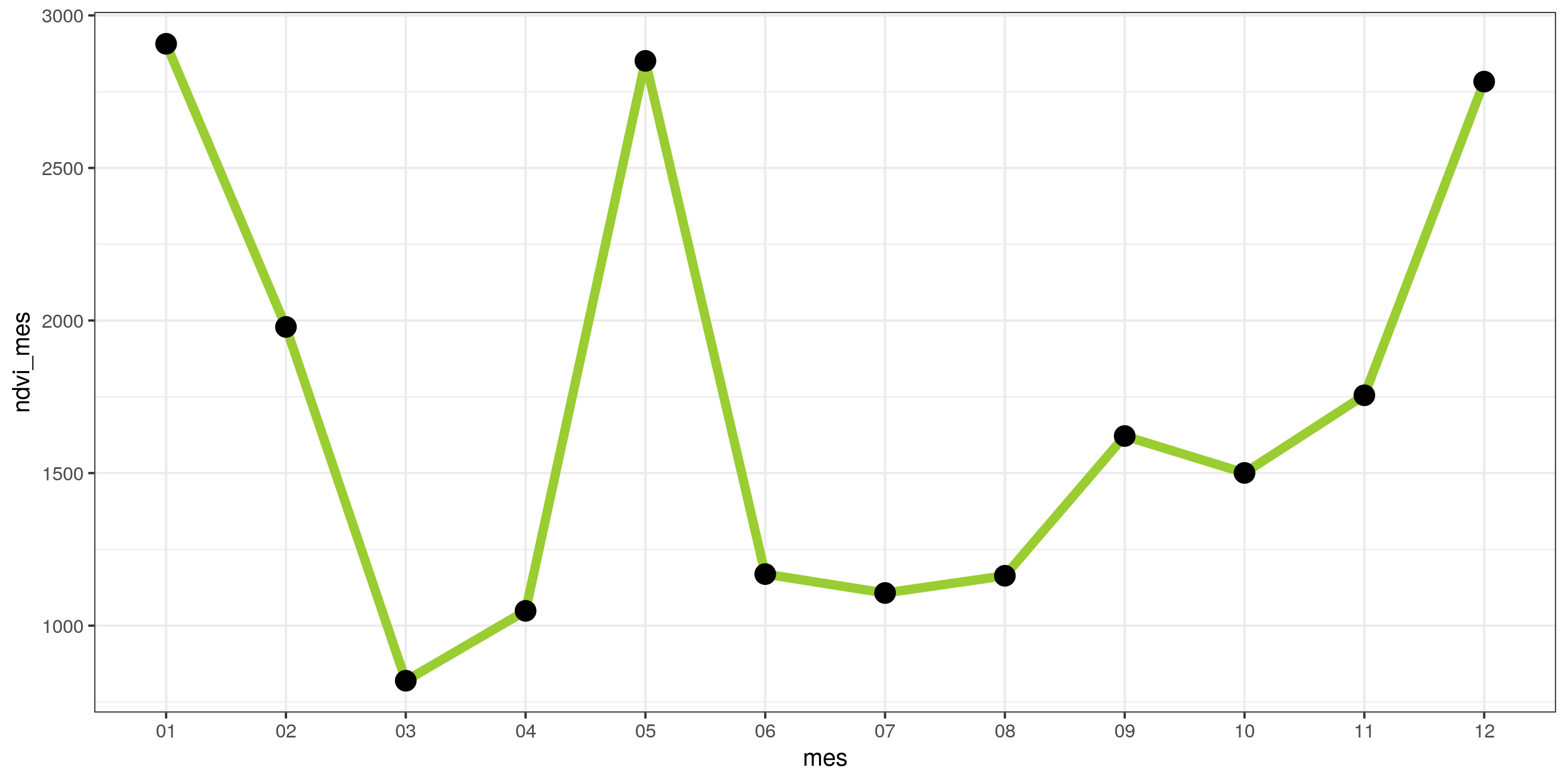
modis\_feb <- ee$Image(list\_modis$filterDate('2016-02-01','2016-02-29')$mean())  
viz <- list(min = 0.0,  
 max = 9000.0,  
 bands = "NDVI",  
 palette = c(  
 'FFFFFF', 'CE7E45',   
 'DF923D', 'F1B555',  
 'FCD163', '99B718',  
 '74A901', '66A000',  
 '529400', '3E8601',   
 '207401', '056201',  
 '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.

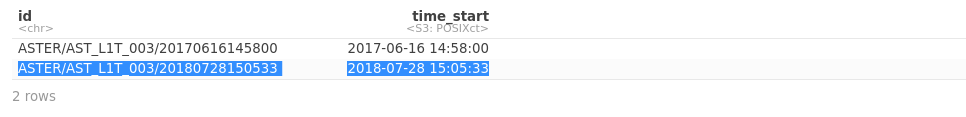
*obs: èe\_extract() nos permite estraer los valores pixeles a la geometría asociada!*

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()



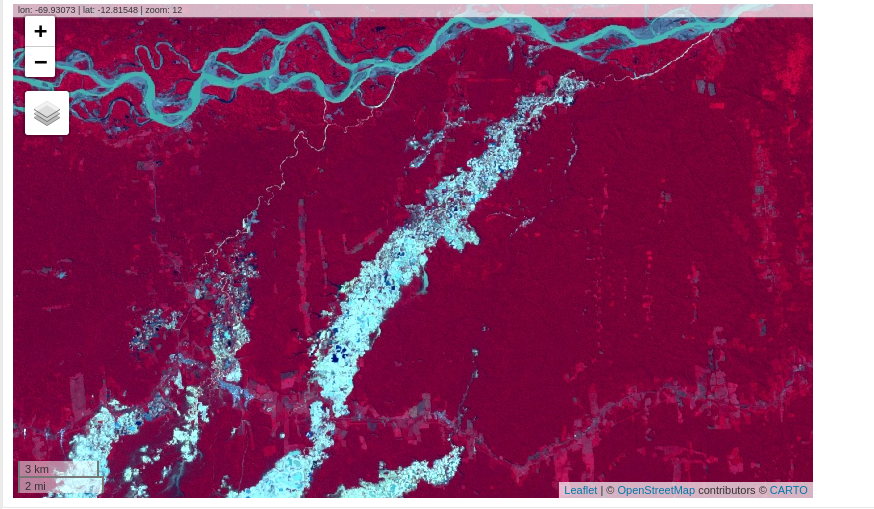
Visualización de imágenes ASTER

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)



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))

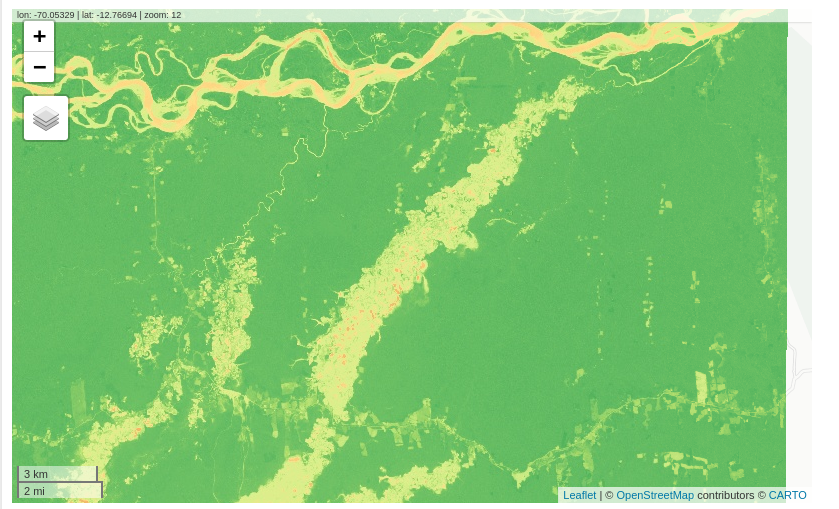


## Cálculo de índices espectrales

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

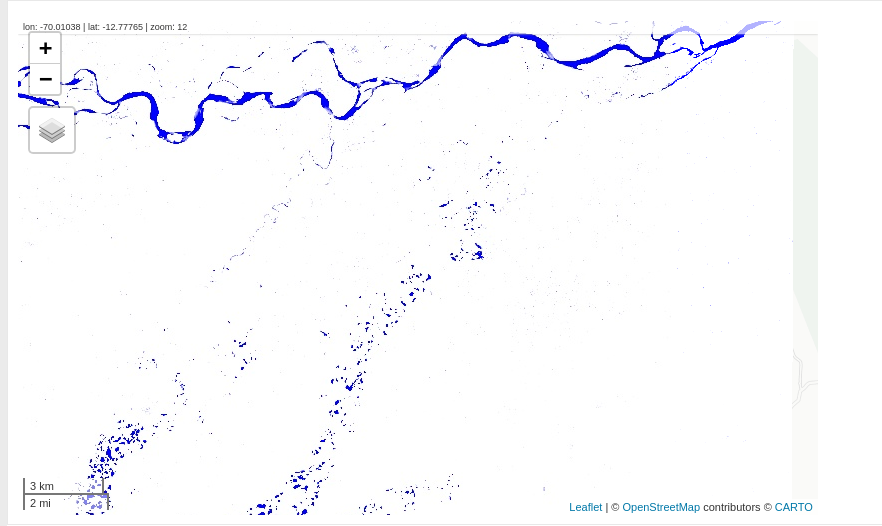
**NDVI en Sentinel2**

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: SAR para el mapeo de descargas máximas y deslizamientos usando rgee

## Obtención de imágenes Sentinel - 1

## Ámbito de estudio

## Visualizando datos de sentinel-1

## Filtro de datos de sentinel-1 por fechas

## Combinación RGB

## Aplicando un filtro de “speckle”

## Diferencia entre un antes y después

## Indentificación de áreas inundadas

## Elaboración de mapas temáticos

## Exportando resultados