

Google Earth Engine en R



Antony Barja

Índice

Introducción a rgee	2
Instalación de rgee y otros	2
Sintaxis básica de rgee	3
Explorando el catálogo de datos de Google Earth Engine	3
Explorando y visualizando imágenes Landsat, Sentinel, MODIS y Aster	4
Cálculo de índices espectrales	11
Caso práctico: SAR para el mapeo de descargas máximas y deslizamientos usando rgee	13
Obtención de imágenes Sentinel - 1	13
Ámbito de estudio	13
Visualizando datos de sentinel-1	13
Filtro de datos de sentinel-1 por fechas	13
Combinación RGB	13
Aplicando un filtro de "speckle"	13
Diferencia entre un antes y después	13
Identificación de áreas inundadas	13
Elaboración de mapas temáticos	13
Exportando resultados	13

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 input y output de datos, la visualización en mapas interactivos, la facil extracción de series de tiempo, el manejo y la visualización de metadatos.

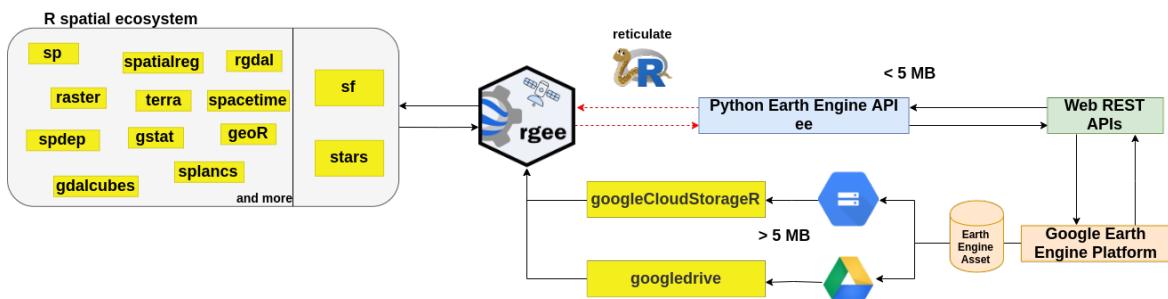


Figura 1: Arquitectura de rgee

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 librerías adicionales, estas son las siguientes:

```
install.packages('mapview')      # Pkgs visualizar de forma interactiva
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 librerías instaladas, empleamos la siguiente función `library() | require()`.

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

Sintaxis 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](#).

JS (Code Editor)	Python	R
<pre>var db = 'CGIAR/SRTM90_V4' var image = ee.Image(db) print(image.bandNames()) #> 'elevation'</pre>	<pre>import ee ee.Initialize() db = 'CGIAR/SRTM90_V4' image = ee.Image(db) image.bandNames(). getInfo() #> [u'elevation']</pre>	<pre>library(rgee) ee_Initialize() db <- 'CGIAR/SRTM90_V4' image <- ee\$Image(db) image\$bandNames()\$getInfo() #> [1] "elevation"</pre>

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

Explorando el catálogo 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	provider
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 fig

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

Visualización del catálogo de Google Earth Engine dentro de R

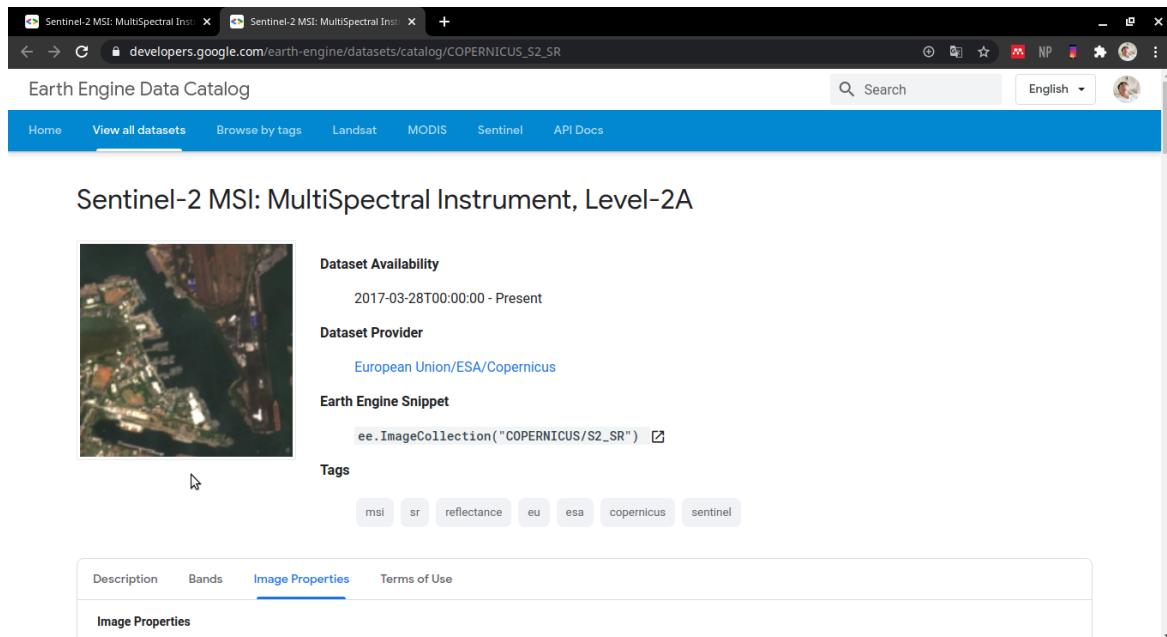


Figura 3: Catálogo de GEE

```
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	start_date	end_date	type
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

Explorando y visualizando imágenes Landsat, Sentinel, MODIS y Aster

```
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 específica:

```
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

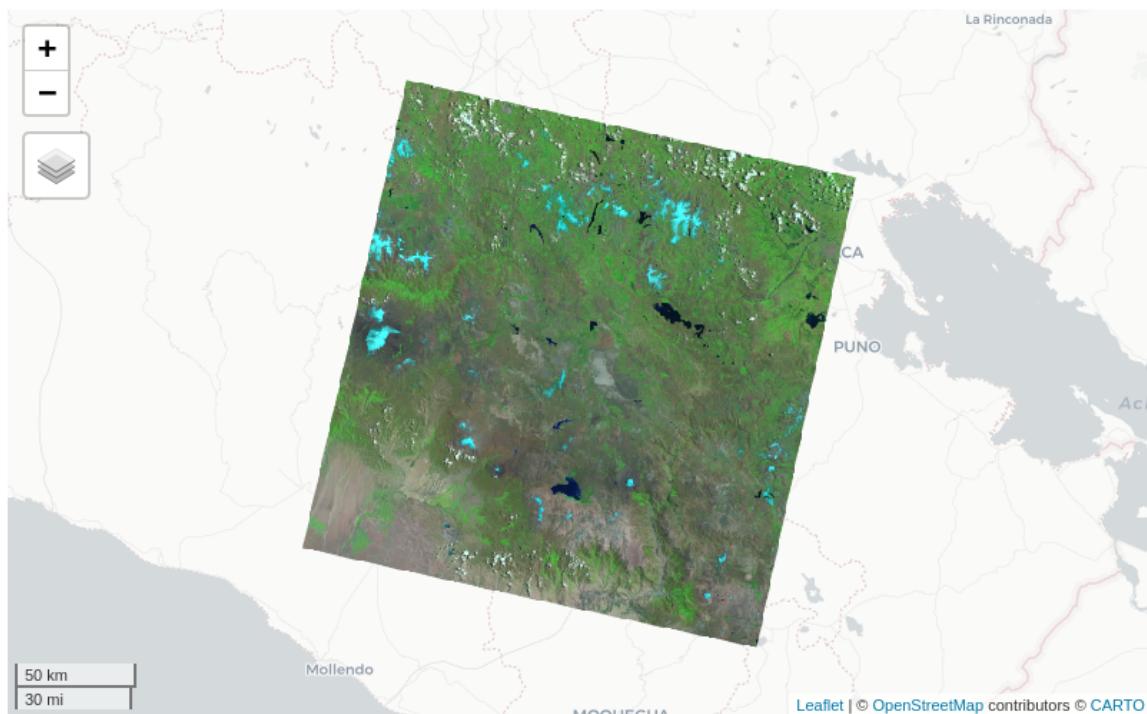
Visualización de la mejor escena:

```
lista <- ee$ImageCollection('LANDSAT/LC08/C01/T1_TOA')$  
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)
```

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(ee$Object = landsat, zoom = 8)  
Map$addLayer(ee$Object = landsat, visParams = viz)
```



Visualización de imágenes sentinel 1, 2, 5

Sentinel-1

```
latlon <- ee$Geometry$Point(-69.96,-12.84)
colecciónVV <- 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'))$  

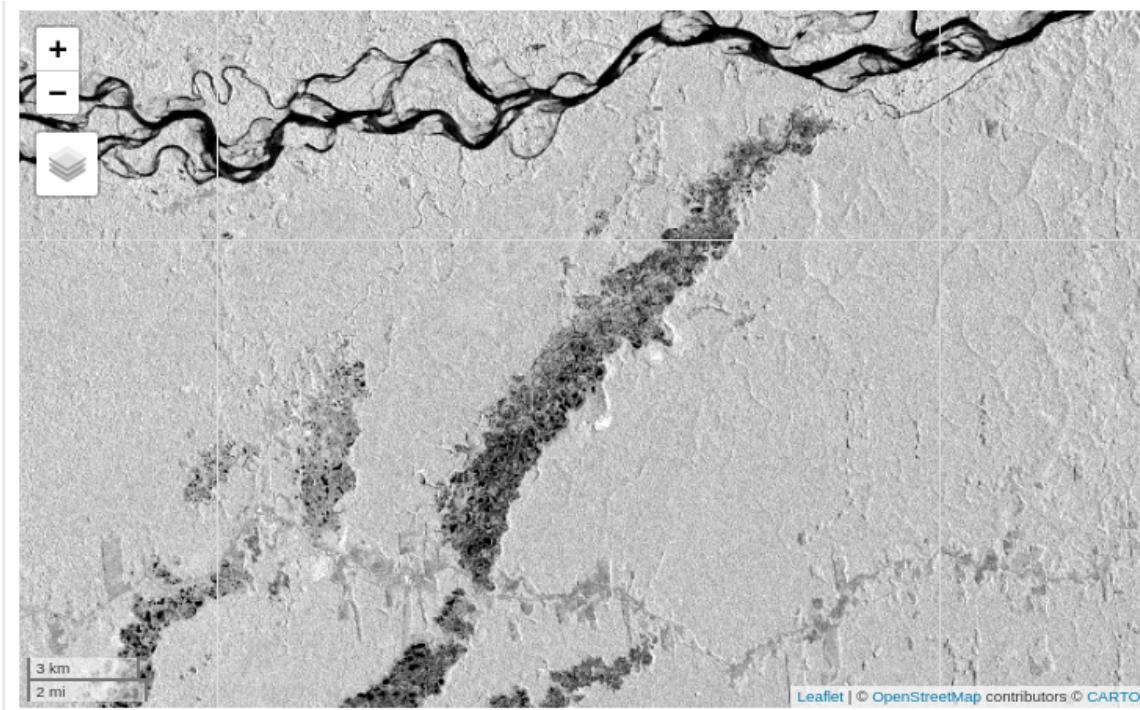
  filterMetadata('resolution_meters', 'equals' , 10)$  

  filterBounds(latlon)$  

  select('VV')

Map$centerObject(latlon,zoom = 12)
colecciónVV$  

  median()%>%
  Map$addLayer(visParams = list(min= -20 , max= -5))
```



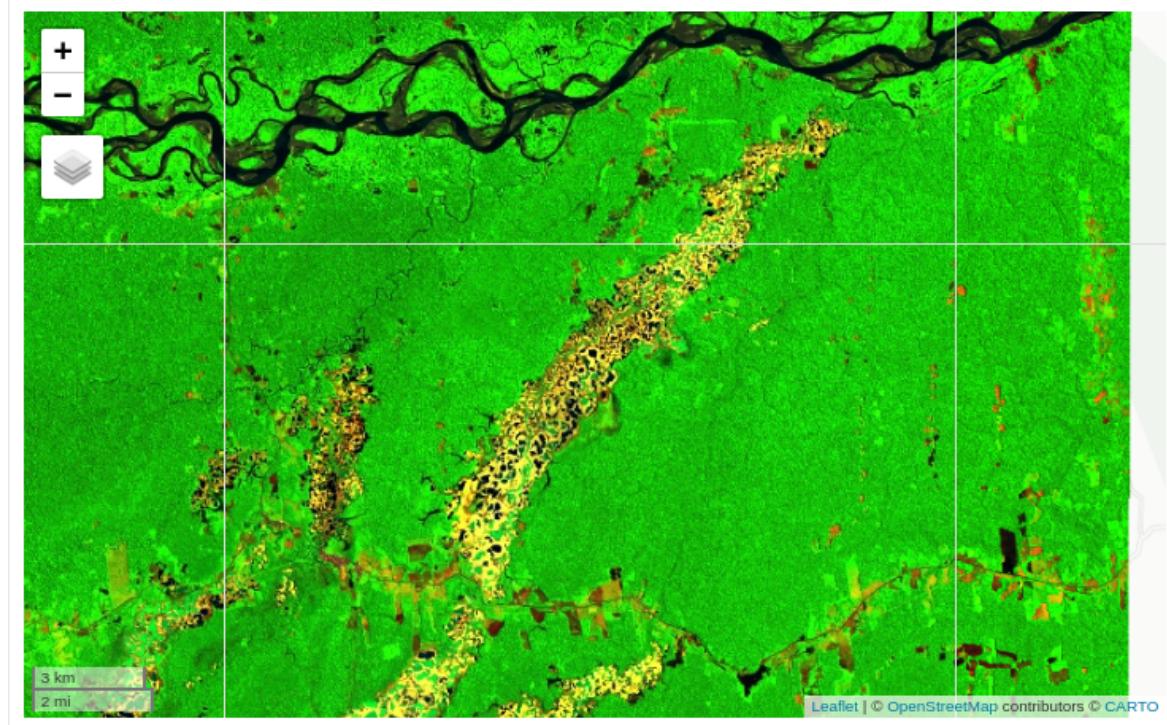
Sentinel-2

```
colección_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(colección_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')),
```



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'))
```



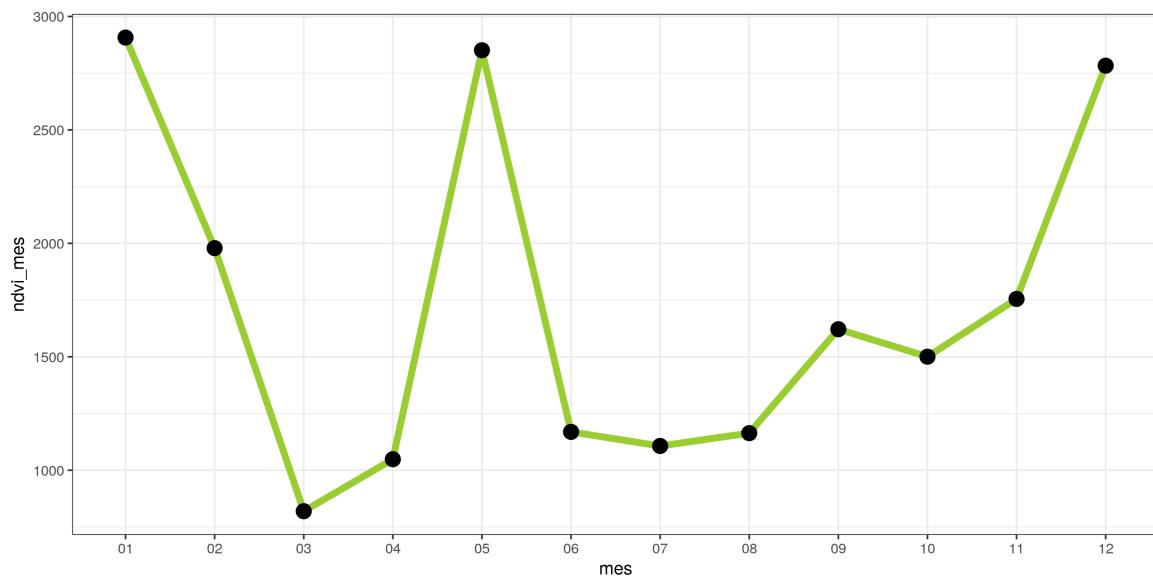
Con rgee puedes analizar **series de tiempos** de forma rápida y con pocas líneas de código, para esta ocasión vamos a ver la variación mensual de nuestro punto de control en campo.

obs: ee_extract() nos permite extraer 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

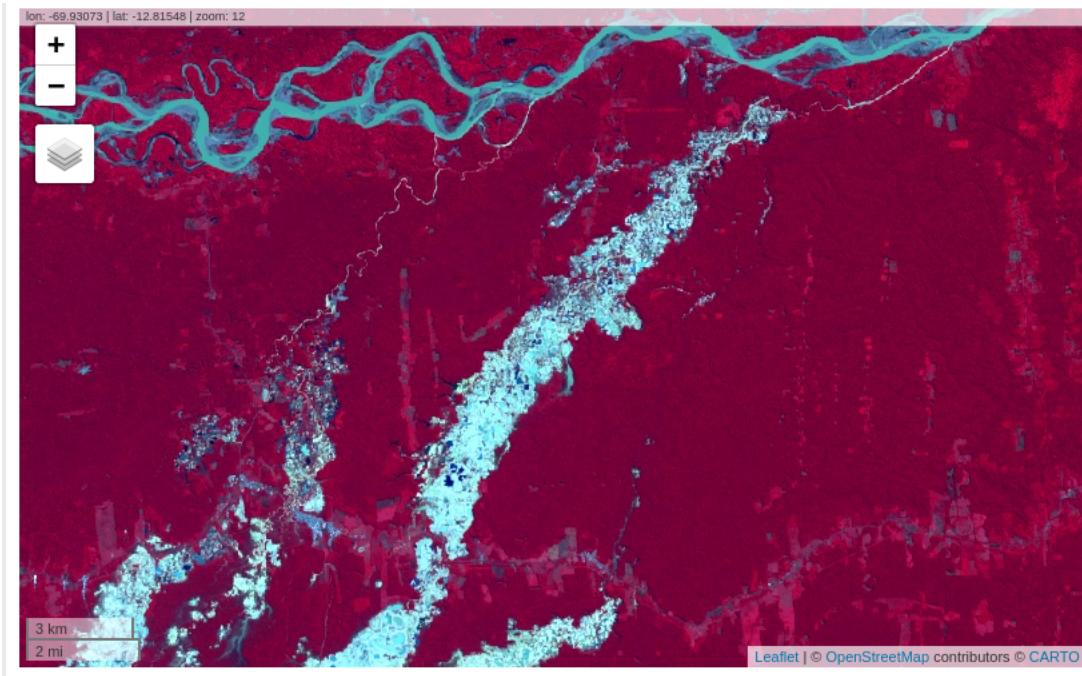
```
list_aster <- ee$ImageCollection('ASTER/AST_L1T_003')$  
filterDate('2016-01-01', '2018-12-15')$  
filterBounds(latlon)$  
filterMetadata('CLOUDCOVER', 'less_than', 1)  
  
ee_get_date_ic(list_aster)
```

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))
```



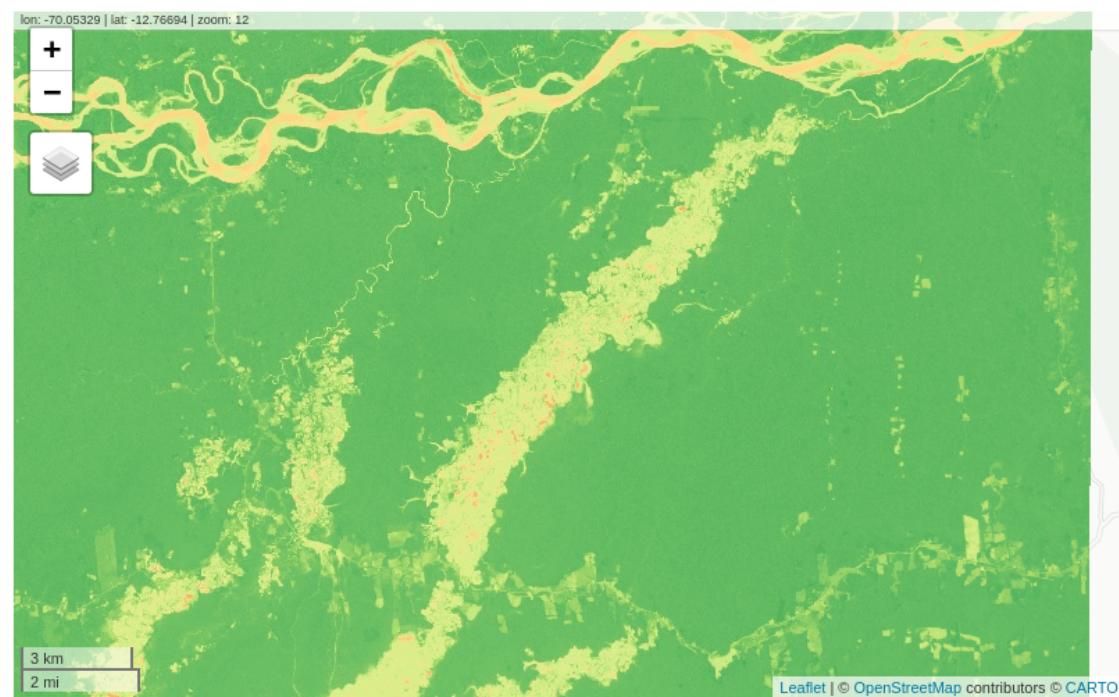
Cálculo de índices espectrales

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

NDVI en Sentinel2

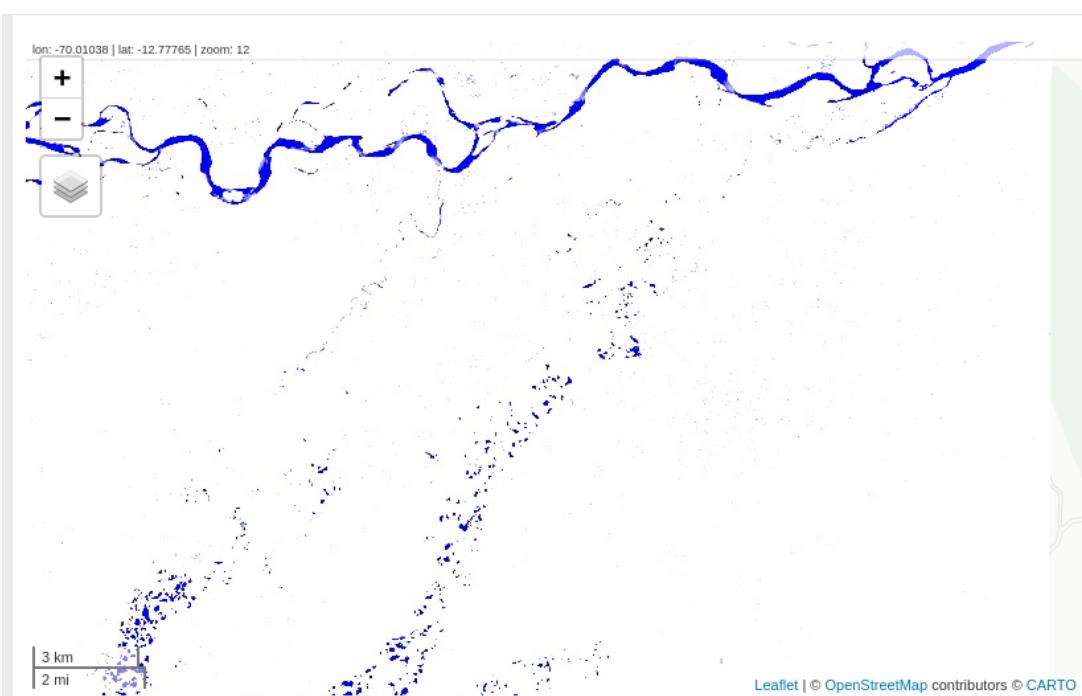
```
Map$centerObject(latlon, zoom = 12)
viz <- list(palette = c(
  "#d73027", "#f46d43",
  "#fdbe61", "#fee08b",
  "#d9ef8b", "#a6d96a",
  "#66bd63", "#1a9850")
)

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



NDWI en Sentinel2

```
viz <- list(  
  min = -0.15,  
  max = 0.65,  
  palette = c(  
    '#fffff', '#fffff', '#fffff',  
    '#fffff', '#fffff', '#0000ff',  
    '#0000ff'))  
)  
  
sen2$normalizedDifference(c('B8','B11')) %>%  
  Map$addLayer(visParams = viz)
```



Caso práctico: SAR para el mapeo de descargas máximas y deslizamientos usando rggee

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