In this blog post I’ll explain the functionality of the PlanetNICFI R package based on a *change detection* use case. The official website of NICFI (Norway’s International Climate and Forest Initiative) includes all the details about the initiative against global deforestation. This initiative was also covered extensively on the web especially from the provider of the free Satellite Imagery. Users have the opportunity to download high-resolution imagery of forests globally using a simple sign up form.

To take advantage of the **PlanetNICFI** R package you will need also an **API key** which you can receive once you are registered. For more details see the Getting Started with Planet APIs website.

The **Documentation** of the R package includes details on how to download and process **monthly** data for an Area of Interest (AOI), however, in this vignette I’ll use **bi-annual** data because it is available since **2015** whereas **monthly** data is available since **September 2020**.

**Change Detection (using bi-annual Imagery)**

To spot deforestation areas we will perform change detection based on a **Sugarcane cultivation area in Bolivia**. For this change detection task we will use two images:

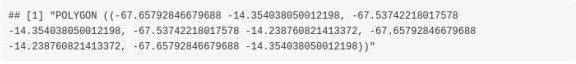
the **first bi-annual** Image of **2016**

the **first bi-annual** Image of **2018**

and the following Well Known Text (WKT) as a cropped area utilizing geojson.io,

wkt\_file = system.file('data\_files/Sugar\_Cane\_Bolivia.wkt', package = "PlanetNICFI") WKT = readLines(wkt\_file, warn = FALSE

WK



First, we have to create a variable and store the **API key** (which a user receives as explained previously),

api\_key = 'use\_your\_planet\_nicfi\_API\_key\_here'

Then we have to extract the **bi-annual mosaics**

require(PlanetNICFI)

mosaic\_files = nicfi\_mosaics(planet\_api\_key = api\_key

type = 'bi\_annually' crs\_bbox = 4326

URL = https://api.planet.com/basemaps/v1/mosaics', verbose = FALSE

dtbl = mosaic\_files$dtbl\_mosaic colnames(dtbl)

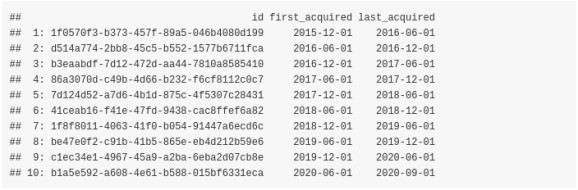


The output of the **nicfi\_mosaics** function is a named list of length 2 and we will keep the data.table (**dtbl\_mosaic**) which includes the **mosaic id’s**

and the **bi-annual Dates** (among other columns) as the previous and the next output shows,

cols\_keep = c('id', 'first\_acquired', 'last\_acquired') dtbl\_keep = dtbl[, ..cols\_keep]

dtbl\_keep



Then we will extract the **mosaic Quads** of the images for years **2016** and **2018**. The Planet Web API for the NICFI data includes both **Quads**

(pixel images of size 4096 x 4096) and **tiles** (pixel images of size 256 x 256) for the global forest areas.

The **PlanetNICFI** R package makes use of **Quads** because it allows the user to extract a specified Area of Interest (AOI) based on a **bounding box**, as I’ll explain later in this vignette.

In the previous subset **dtbl\_keep** data.table we see that

the **first bi-annual** image of **2016** is in **row 1** (**2015-12-01 to 2016-06-01**) and the **first bi-annual** image of **2018** is in **row 5** (**2017-12-01 to 2018-06-01**)

Therefore, we will first extract the first 10 pages for the Quads for the year 2016 (because a maximum of 10 pages is sufficient and will cover our AOI),

index\_year\_2016 = 1

mosaic\_ID\_2016 = dtbl\_keep$id[index\_year\_2016

quad\_files = nicfi\_quads\_bbox(planet\_api\_key = api\_key,

mosaic\_id = mosaic\_ID\_2016 bbox\_AOI = NULL

wkt\_AOI = WKT page\_size = 10

crs\_bbox = 4326 verbose = FALSE

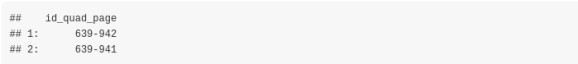
dtbl\_quads = quad\_files$quads colnames(dtbl\_quads)



In the same way as with the *mosaics*, the output of the **nicfi\_quads\_bbox** function is a named list (this time of length 3) and we will keep the data.table (**dtbl\_quads**) which includes the **id\_quad\_page**, **quad\_link\_download** and the **quad\_link\_thumbnail** columns which are required for our analysis.

cols\_keep\_quads = c('id\_quad\_page', 'quad\_link\_download', 'quad\_link\_thumbnail') dtbl\_keep\_quads = dtbl\_quads[, ..cols\_keep\_quads]

dtbl\_keep\_quads[, 'id\_quad\_page']



Although we specified a maximum of 10 pages, as we see from the output, only 2 Quads are returned for our input bounding box.

The **aria2c\_download\_paths** function of the R package allows the user to format the download web URL paths either of the **thumbnail png** or of the bigger **tif** files,

url\_paths\_2016 = aria2c\_download\_paths(mosaic\_output = mosaic\_files,

mosaic\_id = mosaic\_ID\_2016 quads\_output = quad\_files

img\_type = 'thumbnail'

url\_paths\_2016

and we have also to specify a temporary directory so that we can download all required files,

temp\_dir\_out = tempdir() temp\_dir\_out

We set the number of threads to run in parallel equal to the number of the downloaded files and adjust the *aria2c* parameters too,

all\_threads = parallel::detectCores() set\_threads = length(url\_paths\_2016) / 2

num\_threads = ifelse(set\_threads < all\_threads, set\_threads, all\_threads) aria\_args = '--allow-overwrite --file-allocation=none --retry-wait=5 --max-tries=0'

res\_downl = aria2c\_bulk\_donwload(vector\_or\_file\_path = url\_paths\_2016,

default\_directory = temp\_dir\_out threads = num\_threads

verbose = FALSE secondary\_args\_aria = aria\_args

The following images correspond to the downloaded .png’s of the **year 2016** and show the AOI in low resolution,

# Image (a) Image (b)



We can now proceed to download the **.tif** files which are required for the change detection task. We slightly modify the **aria2c\_download\_paths**

function by adjusting the **img\_type** parameter to **‘tif’**. The following code snippet downloads the .tif images of **year 2016**,

url\_paths\_2016\_tif = aria2c\_download\_paths(mosaic\_output = mosaic\_files,

mosaic\_id = mosaic\_ID\_2016 quads\_output = quad\_files img\_type = 'tif'

res\_downl\_tif = aria2c\_bulk\_donwload(vector\_or\_file\_path = url\_paths\_2016\_tif,

default\_directory = temp\_dir\_out threads = num\_threads

verbose = FALSE secondary\_args\_aria = aria\_args

We have 2 images for our bounding box area (each one is *more than 100 MB*), therefore we have to create a *single cropped image* and for this to happen we’ll have to

**create a Virtual Raster** file of the 2 .tif files

**transform the projection** of the bounding box (which is ‘EPSG:4326’) to the projection of the downloaded .tif files

**crop the Virtual Raster** using the transformed bounding box to match our AOI The following code snippet shows the code of the mentioned steps,

#..................................................................

# create a Virtual Raster (VRT) file from the downloaded .tif files

#..................................................................

VRT\_out = file.path(temp\_dir\_out, glue::glue("{mosaic\_ID\_2016}.vrt")) res\_vrt = create\_VRT\_from\_dir(dir\_tifs = temp\_dir\_out,

output\_path\_VRT = VRT\_out file\_extension = '.tif' verbose = TRUE

#.......................................................

# Adjust the Coordinate Reference System of the bounding # box from 4326 to the projection of the .tif files #.......................................................

wkt\_sf = sf::st\_as\_sfc(WKT, crs = 4326

crs\_value = gdalUtils::gdalsrsinfo(VRT\_out, as.CRS = TRUE) proj\_info = crs\_value@projargs

if (is.na(proj\_info)) {

stop("The 'gdalUtils::gdalsrsinfo' returned an NA! 'proj4' is not available in your OS!"

}

wkt\_transf = sf::st\_transform(wkt\_sf, crs = proj\_info bbx\_transf = sf::st\_bbox(wkt\_transf

#..............................................................................

# crop the output .vrt file based on the bounding box and save the output image #..............................................................................

pth\_crop\_out = file.path(temp\_dir\_out, glue::glue("{mosaic\_ID\_2016}\_CROPPED.tif"))

bbx\_crop = list(xmin = as.numeric(bbx\_transf['xmin'] , xmax = as.numeric(bbx\_transf['xmax'] , ymin = as.numeric(bbx\_transf['ymin'] , ymax = as.numeric(bbx\_transf['ymax'] )

warp\_obj = nicfi\_crop\_images(input\_pth = VRT\_out

output\_pth = pth\_crop\_out bbox\_AOI = bbx\_crop threads = num\_threads

of = 'GTiff' resize\_method = 'lanczos' verbose = TRUE

The next image shows the **cropped area for the year 2016** in high resolution,



We can come to the cropped image for the **year 2018** in the same way by adjusting the code in the following way,

index\_year\_2018 = 5

mosaic\_ID\_2018 = dtbl\_keep$id[index\_year\_2018

quad\_files\_2018 = nicfi\_quads\_bbox(planet\_api\_key = api\_key,

mosaic\_id = mosaic\_ID\_2018 bbox\_AOI = NULL

wkt\_AOI = WKT page\_size = 10

crs\_bbox = 4326 verbose = FALSE

url\_paths\_2018 = aria2c\_download\_paths(mosaic\_output = mosaic\_files,

mosaic\_id = mosaic\_ID\_2018 quads\_output = quad\_files\_2018 img\_type = 'tif'

#.....................................................................

# create a new temporary directory to save the .tif files of year 2018 #.....................................................................

temp\_dir\_2018 = file.path(temp\_dir\_out, 'year\_2018')

if (!dir.exists(temp\_dir\_2018)) dir.create(temp\_dir\_2018)

res\_downl\_tif\_2018 = aria2c\_bulk\_donwload(vector\_or\_file\_path = url\_paths\_2018,

default\_directory = temp\_dir\_2018 threads = num\_threads

verbose = FALSE secondary\_args\_aria = aria\_args

VRT\_out\_2018 = file.path(temp\_dir\_2018, glue::glue("{mosaic\_ID\_2018} vrt"))

res\_vrt = create\_VRT\_from\_dir(dir\_tifs = temp\_dir\_2018,

output\_path\_VRT = VRT\_out\_2018 file\_extension = '.tif' verbose = TRUE

pth\_crop\_2018 = file.path(temp\_dir\_2018, glue::glue("{mosaic\_ID\_2018}\_CROPPED.tif"))

warp\_obj = nicfi\_crop\_images(input\_pth = VRT\_out\_2018

output\_pth = pth\_crop\_2018 bbox\_AOI = bbx\_crop threads = num\_threads

of = 'GTiff' resize\_method = 'lanczos' verbose = TRUE

The next image shows the **cropped area for the year 2018**,



Now that we have the cropped images for both years we can perform change detection. One method that can be used directly to the cropped images is **Change Vector Analysis**. I’ll use the Rstoolbox package which includes the **rasterCVA** function. Based on the documentation “… Change Vector Analysis (CVA) is used to identify spectral changes between two identical scenes which were acquired at different times …”. The following code snippet loads the .tif files and computes CVA using the output image of the year 2018 as a **reference**,

orig\_rst = raster::brick(x = pth\_crop\_2018) # image 2018: origin or reference change\_rst = raster::brick(x = pth\_crop\_out) # image 2016

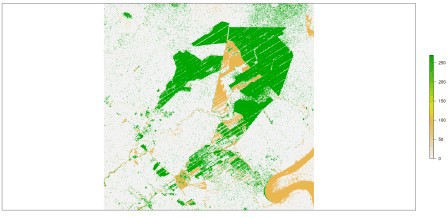
bands = c(1, 5) # bands 1 and 5 depict the difference better cva = RStoolbox::rasterCVA(x = orig\_rst[[bands]], y = change\_rst[[bands]])

To observe the differences in more detail we can arrange the plots side-by-side and add the CVA plot too,

# AOI (2016) AOI (2018)



sp::plot(cva$angle, axes = F)



By solely plotting the **angle** of the **Change Vector Analysis** we can clearly see the deforestation of the area in green color.