NASA Coding Lab

Emma Reich

10/20/2020

```
# Load necessary packages into R
library(getPass)            # A micro-package for reading passwords
```

```
## Warning: package 'getPass' was built under R version 4.0.2
```

```
library(httr)               # To send a request to the server/receive a response from the server
```

```
## Warning: package 'httr' was built under R version 4.0.2
```

```
library(jsonlite)           # Implements a bidirectional mapping between JSON data and the most importa
```

```
## Warning: package 'jsonlite' was built under R version 4.0.2
```

```
library(ggplot2)            # Functions for graphing and mapping
library(tidyr)              # Function for working with tabular data
```

```
## Warning: package 'tidyr' was built under R version 4.0.2
```

```
library(dplyr)              # Function for working with tabular data
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)              # Read rectangular data like CSV

library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.0.2

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:httr':
##
##     config

## The following object is masked from 'package:stats':
```

```
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```r
library(neonUtilities)
source('neon_token_source.R')
soi <- c("SRER","BONA")
```

```r
outDir <- file.path('./NASAdata/')        # Create an output directory if it doesn't exist
suppressWarnings(dir.create(outDir))
```

```r
source('./EARTHDATA_Token.R') #path will change based on where you stored it
exists('user')
```

```
## [1] TRUE
```

```r
secret <- jsonlite::base64_enc(paste(user, password, sep = ":"))  # Encode the string of username and p
```

1. Choose two NEON sites in *different* ecoregions. Then complete the following for **each** of your two NEON sites://

I'll use SRER in D14 Desert Southwest and BONA in D19 Taiga.

2. Using your Earth Data account submit a point-based request to AppEEARS to pull 250m NDVI from AQUA and TERA for 2017, 2018, & 2019.

Hint: How might you decide on the lat/lon to submit? Metadata for the site? NEON TOS data?

```r
API_URL = 'https://lpdaacsvc.cr.usgs.gov/appeears/api/'  # Set the AppEEARS API to a variable
```

```r
# Insert API URL, call login service, set the component of HTTP header, and post the request to the ser
response <- httr::POST(paste0(API_URL,"login"),
                       add_headers("Authorization" = paste("Basic", gsub("\n", "", secret)),
                                   "Content-Type" ="application/x-www-form-urlencoded;charset=UTF-8"),
                       body = "grant_type=client_credentials")

response_content <- content(response)                          # Retrieve the content of the request
token_response <- toJSON(response_content, auto_unbox = TRUE)  # Convert the response to the JSON objec
remove(user, password, secret, response)                       # Remove the variables that are not need
prettify(token_response)                                       # Print the prettified response
```

```
## {
##     "token_type": "Bearer",
##     "token": "AjLSbVpXqgkF2C2r6MHm4l1SomvEff6BJs-5JVX1TgTPPsT3bO1Hbkoq0KSrQe9V0HJHrlQfmE-MNXSe10npFg
##     "expiration": "2020-11-03T19:53:23Z"
## }
##
```

```r
prods_req <- GET(paste0(API_URL, "product"))             # Request the info of all products from produc
prods_content <- content(prods_req)                      # Retrieve the content of request
all_Prods <- toJSON(prods_content, auto_unbox = TRUE)    # Convert the info to JSON object
remove(prods_req, prods_content)                         # Remove the variables that are not needed any
#prettify(all_Prods)                                      # Print the prettified product response
```

```r
# Divides information from each product.
divided_products <- split(fromJSON(all_Prods), seq(nrow(fromJSON(all_Prods))))
# Create a list indexed by the product name and version
```

```r
products <- setNames(divided_products,fromJSON(all_Prods)$ProductAndVersion)
# Print no. products available in AppEEARS
sprintf("AppEEARS currently supports %i products." ,length(products))
```

```
## [1] "AppEEARS currently supports 121 products."
```

```r
NDVI_Products <- list()                                        # Create an empty list
for (p in products){                                          # Loop through the product list
  if (grepl('NDVI', p$Description )){              # Look through the product description for a keyword
    NDVI_Products <- append(NDVI_Products, p$ProductAndVersion) # Append the NDVI products to the list
  }
}


m250_Products <- list()                                        # Create an empty list
for (p in products){                                          # Loop through the product list
  if (grepl('250m', p$Resolution )){              # Look through the product description for a keyword
    m250_Products <- append(m250_Products, p$ProductAndVersion) # Append the NDVI products to the list
  }
}


# NDVI 250m res products
intersect(NDVI_Products, m250_Products)
```

```
## [[1]]
## [1] "MOD13Q1.006"
##
## [[2]]
## [1] "MYD13Q1.006"
##
## [[3]]
## [1] "eMODIS_Smoothed_NDVI.001"
```

```r
  # MOD13Q1.006
  # MYD13Q1.006
```

```r
desired_products <- c("MOD13Q1.006", "MYD13Q1.006")   # Create a vector of desired products
desired_products
```

```
## [1] "MOD13Q1.006" "MYD13Q1.006"
```

```r
# Request layers for the 1st product in the list:
MOD13Q1_req <- GET(paste0(API_URL,"product/", desired_products[1]))  # Request the info of a product fr
MOD13Q1_content <- content(MOD13Q1_req)                             # Retrieve content of the request
MOD13Q1_response <- toJSON(MOD13Q1_content, auto_unbox = TRUE)      # Convert the content to JSON objec
remove(MOD13Q1_req, MOD13Q1_content)                                # Remove the variables that are not
#prettify(MOD13Q1_response)                                          # Print the prettified response
names(fromJSON(MOD13Q1_response))                                    # print the layer's names
```

```
##  [1] "_250m_16_days_EVI"
##  [2] "_250m_16_days_MIR_reflectance"
##  [3] "_250m_16_days_NDVI"
##  [4] "_250m_16_days_NIR_reflectance"
##  [5] "_250m_16_days_VI_Quality"
##  [6] "_250m_16_days_blue_reflectance"
##  [7] "_250m_16_days_composite_day_of_the_year"
##  [8] "_250m_16_days_pixel_reliability"
##  [9] "_250m_16_days_red_reflectance"
## [10] "_250m_16_days_relative_azimuth_angle"
## [11] "_250m_16_days_sun_zenith_angle"
## [12] "_250m_16_days_view_zenith_angle"
```

```
desired_layers <- c("_250m_16_days_NDVI")   # Create a vector of desired layers
desired_prods <- desired_products  # Create a vector of products including the desired layers
# Create a data frame including the desired data products and layers
layers <- data.frame(product = desired_prods, layer = desired_layers)
```

```
# Lat/lon values for SRER and BONA
# These are the tower locations, since we will be looking at Phenocam values later

lat_s <- 31.91068
long_s <-  -110.83549


lat_b <- 65.15401
long_b <- -147.50258
```

First I'll do a point request for SRER:

```
startDate <- "01-01-2017"       # Start of the date range for  which to extract data: MM-DD-YYYY
endDate <- "12-31-2019"         # End of the date range for  which to extract data: MM-DD-YYYY
recurring <- FALSE             # Specify True for a recurring date range
#yearRange <- c(2017,2019)      # If recurring = True, set yearRange, change start/end date to MM-DD

lat <- c(lat_s, lat_b)        # Latitude of the point sites
lon <- c(long_s, long_b)    # Longitude of the point sites
id <- c("0")                      # ID for the point sites
category <- c("SRER", "BONA") # Category for point sites

taskName <- 'NEON SRER NDVI'            # Enter name of the task here
# Whoops, forgot to list BONA in the taskName too

taskType <- 'point'                    # Specify the task type, it can be either "area" or "point"
```

```
# Create a data frame including the date range for the request
date <- data.frame(startDate = startDate, endDate = endDate)
# Create a data frame including lat and long coordinates. ID and category name is optional.
coordinates <- data.frame(id = id, longitude = lon, latitude = lat, category = category)
task_info <- list(date, layers, coordinates)               # Create a list of data frames
names(task_info) <- c("dates", "layers", "coordinates")   # Assign names
task <- list(task_info, taskName, taskType)                # Create a nested list
names(task) <- c("params", "task_name", "task_type")      # Assign names
remove(date, layers, coordinates, task_info)
```

```
task_json <- toJSON(task,auto_unbox = TRUE)   # Convert to JSON object
```

```
token <- paste("Bearer", fromJSON(token_response)$token)      # Save login token to a variable
```

```
# Post the point request to the API task service
response <- POST(paste0(API_URL, "task"),
                 body = task_json ,
                 encode = "json",
                 add_headers(Authorization = token, "Content-Type" = "application/json"))

task_content <- content(response)                                 # Retrieve content of the request
task_response <- prettify(toJSON(task_content, auto_unbox = TRUE))# Convert the content to JSON object
remove(response, task_content)                                    # Remove the variables that are not n
task_response                                                     # Print the prettified task response
```

```
params <- list(limit = 2, pretty = TRUE)                            # Set up query parameters
# Request the task status of last 2 requests from task URL
response_req <- GET(paste0(API_URL,"task"), query = params, add_headers(Authorization = token))
response_content <- content(response_req)                           # Retrieve content of the request
status_response <- toJSON(response_content, auto_unbox = TRUE)      # Convert the content to JSON objec
remove(response_req, response_content)                              # Remove the variables that are not
prettify(status_response)                                           # Print the prettified response
```

```
task_id <- fromJSON(task_response)$task_id                 # Extract the task_id of submitted point req
# Request the task status of a task with the provided task_id from task URL
status_req <- GET(paste0(API_URL,"task/", task_id), add_headers(Authorization = token))
status_content <- content(status_req)                       # Retrieve content of the request
statusResponse <-toJSON(status_content, auto_unbox = TRUE)  # Convert the content to JSON object
stat <- fromJSON(statusResponse)$status                     # Assign the task status to a variable
remove(status_req, status_content)                          # Remove the variables that are not needed
prettify(statusResponse)                                    # Print the prettified response
```

```
while (stat != 'done') {
  Sys.sleep(5)
  # Request the task status and retrieve content of request from task URL
  stat_content <- content(GET(paste0(API_URL,"task/", task_id), add_headers(Authorization = token)))
  stat <-fromJSON(toJSON(stat_content, auto_unbox = TRUE))$status    # Get the status
  remove(stat_content)
  print(stat)
}
```

```
# Request the task bundle info from API bundle URL
response <- GET(paste0(API_URL, "bundle/", task_id), add_headers(Authorization = token))
response_content <- content(response)                          # Retrieve content of the request
bundle_response <- toJSON(response_content, auto_unbox = TRUE)  # Convert the content to JSON object
prettify(bundle_response)                                       # Print the prettified response
```

```
bundle <- fromJSON(bundle_response)$files
for (id in bundle$file_id){
  # retrieve the filename from the file_id
  filename <- bundle[bundle$file_id == id,]$file_name
  # create a destination directory to store the file in
  filepath <- paste(outDir,filename, sep = "/")
  suppressWarnings(dir.create(dirname(filepath)))
  # write the file to disk using the destination directory and file name
  response <- GET(paste0(API_URL, "bundle/", task_id, "/", id),
                  write_disk(filepath, overwrite = TRUE),
                  progress(),
                  add_headers(Authorization = token))
  }
```

```r
params <- list(limit = 6, offset = 20, pretty = TRUE)     # Set up the query parameters
q_req <- GET(paste0(API_URL, "quality"), query = params)  # Request the quality info from quality API_U
q_content <- content(q_req)                               # Retrieve the content of request
q_response <- toJSON(q_content, auto_unbox = TRUE)        # Convert the info to JSON object
remove(params, q_req, q_content)                          # Remove the variables that are not needed
prettify(q_response)                                      # Print the prettified quality information
```

```r
productAndVersion <- 'MOD13Q1.006'                            # Assign ProductAndVersion to a variable
# Request the quality info from quality API for a specific product
MOD13Q1_req <- GET(paste0(API_URL, "quality/", productAndVersion))
MOD13Q1_content <- content(MOD13Q1_req)                      # Retrieve the content of request
MOD13Q1_quality <- toJSON(MOD13Q1_content, auto_unbox = TRUE)# Convert the info to JSON object
remove(MOD13Q1_req, MOD13Q1_content)                         # Remove the variables that are not needed
prettify(MOD13Q1_quality)                                     # Print the prettified quality informatio
```

```
## [
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "Layer": "_250m_16_days_EVI",
##         "QualityProductAndVersion": "MOD13Q1.006",
##         "QualityLayers": [
##             "_250m_16_days_VI_Quality"
##         ],
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "Layer": "_250m_16_days_NDVI",
##         "QualityProductAndVersion": "MOD13Q1.006",
##         "QualityLayers": [
##             "_250m_16_days_VI_Quality"
##         ],
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "Layer": "_250m_16_days_NIR_reflectance",
##         "QualityProductAndVersion": "MOD13Q1.006",
##         "QualityLayers": [
##             "_250m_16_days_pixel_reliability"
##         ],
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "Layer": "_250m_16_days_blue_reflectance",
##         "QualityProductAndVersion": "MOD13Q1.006",
##         "QualityLayers": [
##             "_250m_16_days_pixel_reliability"
##         ],
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##          "Layer": "_250m_16_days_MIR_reflectance",
##          "QualityProductAndVersion": "MOD13Q1.006",
##          "QualityLayers": [
##              "_250m_16_days_pixel_reliability"
##          ],
##          "VisibleToWorker": true
##      },
##      {
##          "ProductAndVersion": "MOD13Q1.006",
##          "Layer": "_250m_16_days_red_reflectance",
##          "QualityProductAndVersion": "MOD13Q1.006",
##          "QualityLayers": [
##              "_250m_16_days_pixel_reliability"
##          ],
##          "VisibleToWorker": true
##      }
## ]
##
```

```
quality_layer <- '_250m_16_days_VI_Quality'                                # assign a quality layer to
# Request the specified quality layer info from quality API
quality_req <- GET(paste0(API_URL, "quality/",  productAndVersion, "/", quality_layer, sep = ""))
quality_content <- content(quality_req)                        # Retrieve the content of request
quality_response <- toJSON(quality_content, auto_unbox = TRUE) # Convert the info to JSON object
remove(quality_req, quality_content)                           # Remove the variables that are not need
prettify(quality_response)                                     # Print the quality response as a data f
```

```
## [
##      {
##          "ProductAndVersion": "MOD13Q1.006",
##          "QualityLayer": "_250m_16_days_VI_Quality",
##          "Name": "MODLAND",
##          "Value": 0,
##          "Description": "VI produced with good quality",
##          "Acceptable": true
##      },
##      {
##          "ProductAndVersion": "MOD13Q1.006",
##          "QualityLayer": "_250m_16_days_VI_Quality",
##          "Name": "MODLAND",
##          "Value": 1,
##          "Description": "VI produced, but check other QA",
##          "Acceptable": false
##      },
##      {
##          "ProductAndVersion": "MOD13Q1.006",
##          "QualityLayer": "_250m_16_days_VI_Quality",
##          "Name": "MODLAND",
##          "Value": 2,
##          "Description": "Pixel produced, but most probably cloudy",
##          "Acceptable": false
##      },
##      {
##          "ProductAndVersion": "MOD13Q1.006",
##          "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "MODLAND",
##         "Value": 3,
##         "Description": "Pixel not produced due to other reasons than clouds",
##         "Acceptable": false
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 0,
##         "Description": "Highest quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 1,
##         "Description": "Lower quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 2,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 3,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 4,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 5,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 6,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 7,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 8,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 9,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 10,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 11,
##         "Description": "Decreasing quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 12,
##         "Description": "Lowest quality",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 13,
##         "Description": "Quality so low that it is not useful",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 14,
##         "Description": "L1B data faulty",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "VI Usefulness",
##         "Value": 15,
##         "Description": "Not useful for any other reason/not processed",
##         "Acceptable": {
```

```
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Aerosol Quantity",
##         "Value": 0,
##         "Description": "Climatology",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Aerosol Quantity",
##         "Value": 1,
##         "Description": "Low",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Aerosol Quantity",
##         "Value": 2,
##         "Description": "Average",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Aerosol Quantity",
##         "Value": 3,
##         "Description": "High",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Adjacent cloud detected",
##         "Value": 0,
##         "Description": "No",
##         "Acceptable": {
##
##         }
##     },
##     {
```

```
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Adjacent cloud detected",
##         "Value": 1,
##         "Description": "Yes",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Atmosphere BRDF Correction",
##         "Value": 0,
##         "Description": "No",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Atmosphere BRDF Correction",
##         "Value": 1,
##         "Description": "Yes",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Mixed Clouds",
##         "Value": 0,
##         "Description": "No",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Mixed Clouds",
##         "Value": 1,
##         "Description": "Yes",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Land/Water Mask",
##         "Value": 0,
##         "Description": "Shallow ocean",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Land/Water Mask",
##         "Value": 1,
##         "Description": "Land (Nothing else but land)",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Land/Water Mask",
##         "Value": 2,
##         "Description": "Ocean coastlines and lake shorelines",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Land/Water Mask",
##         "Value": 3,
##         "Description": "Shallow inland water",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Land/Water Mask",
##         "Value": 4,
##         "Description": "Ephemeral water",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Land/Water Mask",
##         "Value": 5,
##         "Description": "Deep inland water",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Land/Water Mask",
##         "Value": 6,
##         "Description": "Moderate or continental ocean",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Land/Water Mask",
##         "Value": 7,
##         "Description": "Deep ocean",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Possible snow/ice",
##         "Value": 0,
##         "Description": "No",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Possible snow/ice",
##         "Value": 1,
##         "Description": "Yes",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Possible shadow",
##         "Value": 0,
##         "Description": "No",
##         "Acceptable": {
##
##         }
##     },
##     {
##         "ProductAndVersion": "MOD13Q1.006",
##         "QualityLayer": "_250m_16_days_VI_Quality",
##         "Name": "Possible shadow",
##         "Value": 1,
##         "Description": "Yes",
##         "Acceptable": {
##
##         }
##     }
## ]
##
```

```
quality_value <- 0                        # Assign a quality value to a variable
# Request and retrieve information for provided quality value from quality API URL
response <- content(GET(paste0(API_URL, "quality/", productAndVersion, "/", quality_layer, "/", quality_
q_response <- toJSON(response, auto_unbox = TRUE)     # Convert the info to JSON object
remove(response)                                      # Remove the variables that are not needed anymor
prettify(q_response)                                  # Print the prettified response
```

```
## {
##     "Binary Representation": "0b0000000000000000",
##     "MODLAND": {
##         "bits": "0b00",
##         "description": "VI produced with good quality"
##     },
##     "VI Usefulness": {
##         "bits": "0b0000",
##         "description": "Highest quality"
##     },
##     "Aerosol Quantity": {
##         "bits": "0b00",
##         "description": "Climatology"
##     },
##     "Adjacent cloud detected": {
##         "bits": "0b0",
##         "description": "No"
##     },
##     "Atmosphere BRDF Correction": {
##         "bits": "0b0",
##         "description": "No"
##     },
##     "Mixed Clouds": {
##         "bits": "0b0",
##         "description": "No"
##     },
##     "Land/Water Mask": {
##         "bits": "0b000",
##         "description": "Shallow ocean"
##     },
##     "Possible snow/ice": {
##         "bits": "0b0",
##         "description": "No"
##     },
##     "Possible shadow": {
##         "bits": "0b0",
##         "description": "No"
##     }
## }
##
```

```r
# Read the MOD13Q1 results
dfmod <- read_csv("./NASAdata/neon-srer-ndvi/NEON-SRER-NDVI-MOD13Q1-006-results.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ID = col_double(),
##   Latitude = col_double(),
##   Longitude = col_double(),
##   Date = col_date(format = ""),
##   MOD13Q1_006_Line_Y_250m = col_double(),
##   MOD13Q1_006_Sample_X_250m = col_double(),
##   MOD13Q1_006__250m_16_days_NDVI = col_double(),
##   MOD13Q1_006__250m_16_days_VI_Quality = col_double()
## )

## See spec(...) for full column specifications.
```

```r
# Read the MYD13Q1 results
dfmyd <- read_csv("./NASAdata/neon-srer-ndvi/NEON-SRER-NDVI-MYD13Q1-006-results.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ID = col_double(),
##   Latitude = col_double(),
##   Longitude = col_double(),
##   Date = col_date(format = ""),
##   MYD13Q1_006_Line_Y_250m = col_double(),
##   MYD13Q1_006_Sample_X_250m = col_double(),
##   MYD13Q1_006__250m_16_days_NDVI = col_double(),
##   MYD13Q1_006__250m_16_days_VI_Quality = col_double()
## )
## See spec(...) for full column specifications.
```

3. Use QA/QC values to filter out 'poor quality'.

Hint: Look at the QA/QC files that accompany your request

```r
# Look at all column names
names(dfmod)
```

```
##  [1] "Category"
##  [2] "ID"
##  [3] "Latitude"
##  [4] "Longitude"
##  [5] "Date"
##  [6] "MODIS_Tile"
##  [7] "MOD13Q1_006_Line_Y_250m"
##  [8] "MOD13Q1_006_Sample_X_250m"
##  [9] "MOD13Q1_006__250m_16_days_NDVI"
## [10] "MOD13Q1_006__250m_16_days_VI_Quality"
## [11] "MOD13Q1_006__250m_16_days_VI_Quality_bitmask"
## [12] "MOD13Q1_006__250m_16_days_VI_Quality_MODLAND"
## [13] "MOD13Q1_006__250m_16_days_VI_Quality_MODLAND_Description"
## [14] "MOD13Q1_006__250m_16_days_VI_Quality_VI_Usefulness"
## [15] "MOD13Q1_006__250m_16_days_VI_Quality_VI_Usefulness_Description"
## [16] "MOD13Q1_006__250m_16_days_VI_Quality_Aerosol_Quantity"
## [17] "MOD13Q1_006__250m_16_days_VI_Quality_Aerosol_Quantity_Description"
## [18] "MOD13Q1_006__250m_16_days_VI_Quality_Adjacent_cloud_detected"
## [19] "MOD13Q1_006__250m_16_days_VI_Quality_Adjacent_cloud_detected_Description"
## [20] "MOD13Q1_006__250m_16_days_VI_Quality_Atmosphere_BRDF_Correction"
## [21] "MOD13Q1_006__250m_16_days_VI_Quality_Atmosphere_BRDF_Correction_Description"
## [22] "MOD13Q1_006__250m_16_days_VI_Quality_Mixed_Clouds"
## [23] "MOD13Q1_006__250m_16_days_VI_Quality_Mixed_Clouds_Description"
## [24] "MOD13Q1_006__250m_16_days_VI_Quality_Land/Water_Mask"
## [25] "MOD13Q1_006__250m_16_days_VI_Quality_Land/Water_Mask_Description"
## [26] "MOD13Q1_006__250m_16_days_VI_Quality_Possible_snow/ice"
## [27] "MOD13Q1_006__250m_16_days_VI_Quality_Possible_snow/ice_Description"
## [28] "MOD13Q1_006__250m_16_days_VI_Quality_Possible_shadow"
## [29] "MOD13Q1_006__250m_16_days_VI_Quality_Possible_shadow_Description"
```

```r
# Look at all possible modland qaulity descriptions
unique(dfmod$MOD13Q1_006__250m_16_days_VI_Quality_MODLAND_Description)
```

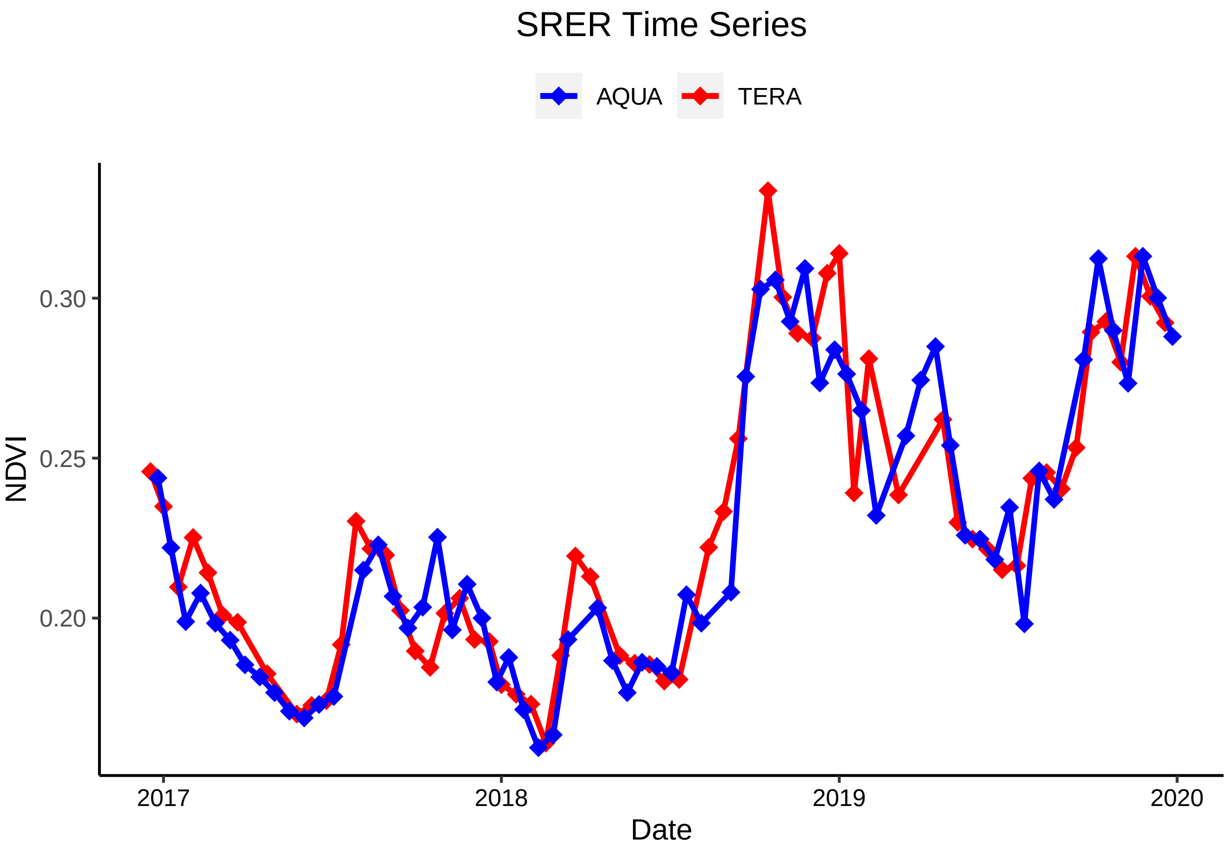
```
## [1] "Pixel not produced due to other reasons than clouds"
## [2] "VI produced, but check other QA"
## [3] "VI produced with good quality"
## [4] "Pixel produced, but most probably cloudy"
```

```r
# For this assignment I'll use 'VI produced with good quality'
# Which is both 'Highest' and 'Lower' quality data according to the VI_Usefulness_Description
dfmod_QC <- dfmod %>%
  filter( MOD13Q1_006__250m_16_days_VI_Quality_MODLAND_Description =='VI produced with good quality')

dfmyd_QC <- dfmyd %>%
  filter( MYD13Q1_006__250m_16_days_VI_Quality_MODLAND_Description =='VI produced with good quality')
```

```r
# I'll also filter by site at this time

dfmod_SRER <- dfmod_QC %>%
  filter( Category =='SRER')

dfmod_BONA <- dfmod_QC %>%
  filter( Category =='BONA')

dfmyd_SRER <- dfmyd_QC %>%
  filter( Category =='SRER')

dfmyd_BONA <- dfmyd_QC %>%
  filter( Category =='BONA')
```

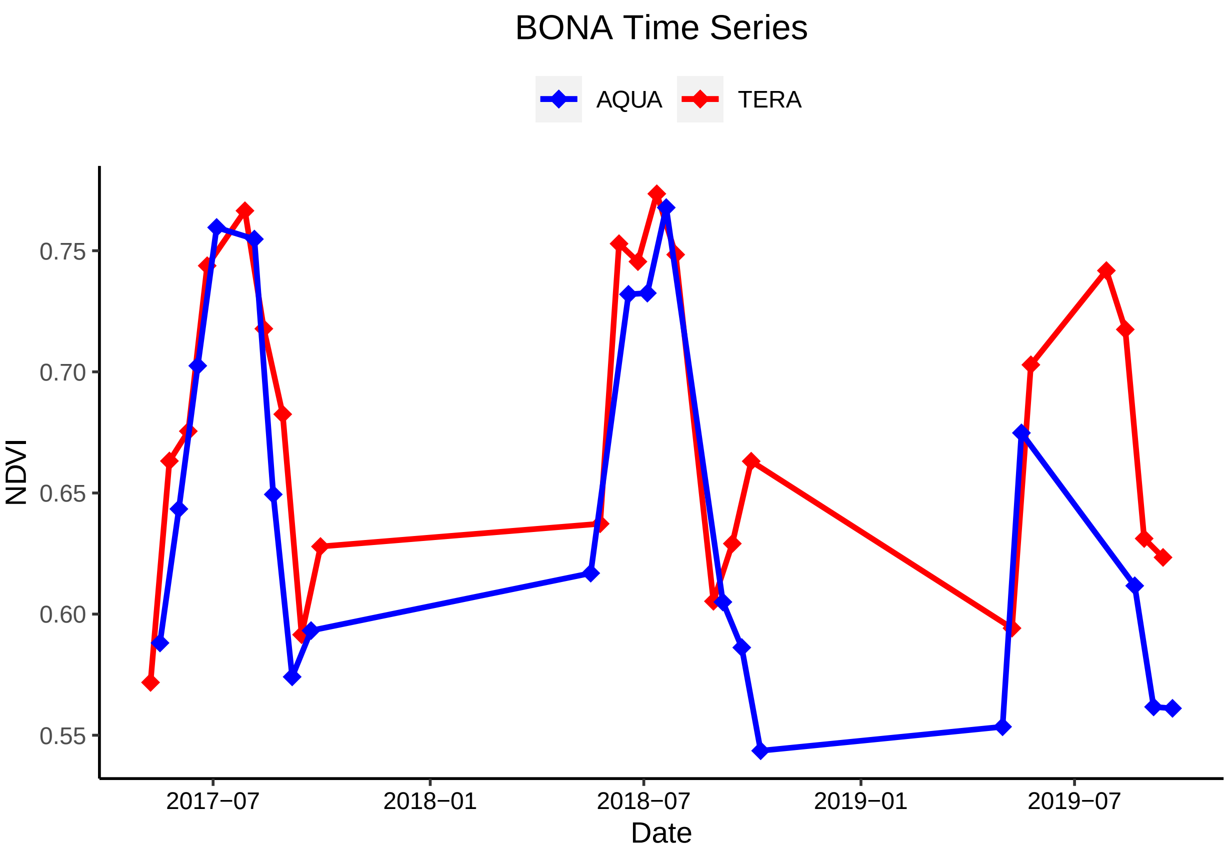
4. Plot 3 years of NDVI from MODIS AQUA and TERA as a timeseries.

```r
# SRER
ggplot()+
  geom_line(data = dfmod_SRER,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA"), size=1)
  geom_point(data = dfmod_SRER,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA"), shape=
  geom_line(data = dfmyd_SRER,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"), size=1)
  geom_point(data = dfmyd_SRER,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"), shape=
  scale_color_manual(name = NULL, values=c("blue", "red")) +
  labs(title = "SRER Time Series", x = "Date", y = "NDVI")+
  theme(legend.position = "top",
        panel.background = element_rect(fill="white"),
        axis.line = element_line(color = "black"),
        axis.text.x = element_text(colour="black"),
        plot.title = element_text(hjust = 0.5))
```



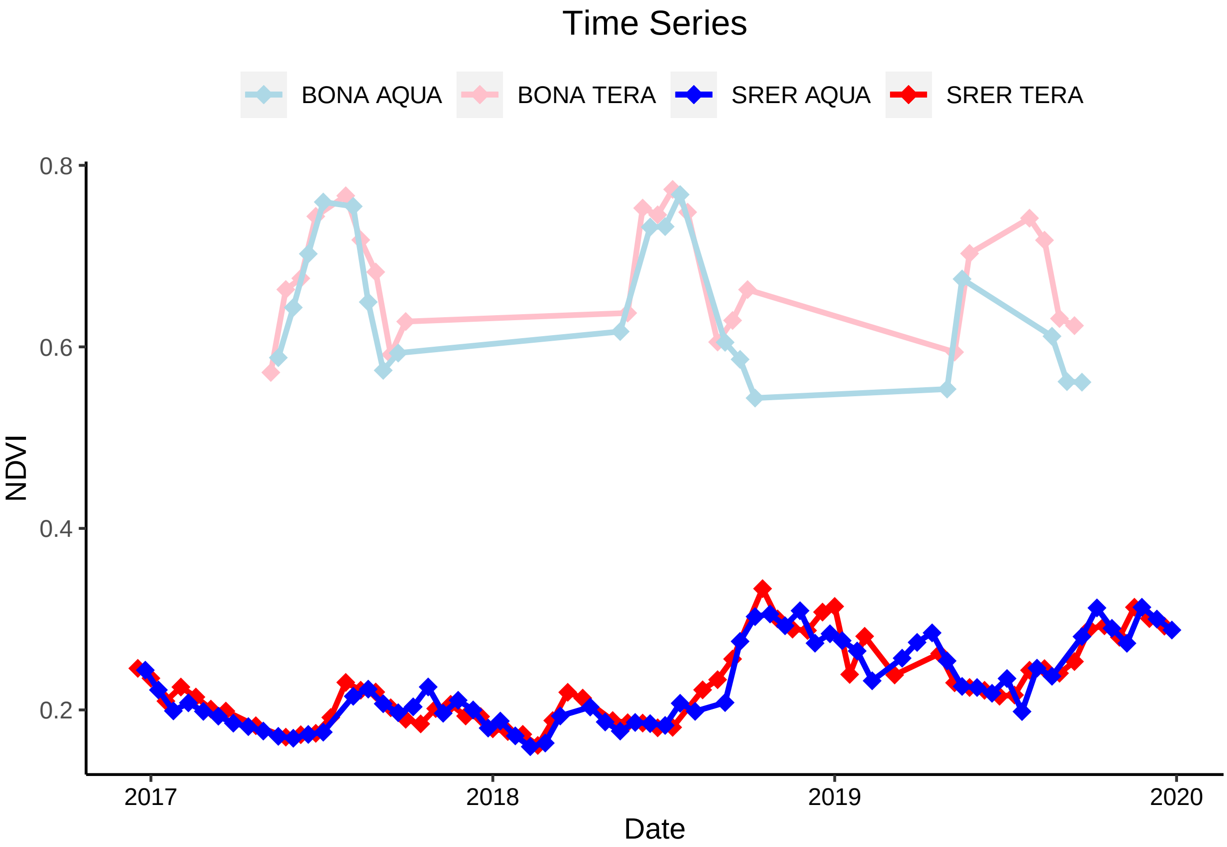
That's really weird! NDVI for SRER seems highest later than I would have thought. SRER might have a high presence of winter annual grasses that dry out during the summer months.

```
# BONA

# Plot both sites
ggplot()+
  geom_line(data = dfmod_BONA,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA"), size=1)
  geom_point(data = dfmod_BONA,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA"), shape=
  geom_line(data = dfmyd_BONA,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"), size=1)
  geom_point(data = dfmyd_BONA,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"), shape=
  scale_color_manual(name = NULL, values=c("blue", "red")) +
  labs(title = "BONA Time Series", x = "Date", y = "NDVI")+
  theme(legend.position = "top",
        panel.background = element_rect(fill="white"),
        axis.line = element_line(color = "black"),
        axis.text.x = element_text(colour="black"),
        plot.title = element_text(hjust = 0.5))
```



That’s interesting, maybe I should plot these two sites together.

```
# Plot both sites
ggplot()+
  geom_line(data = dfmod_SRER,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "SRER TERA"), si
  geom_point(data = dfmod_SRER,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "SRER TERA"), sh
  geom_line(data = dfmyd_SRER,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "SRER AQUA"), si
  geom_point(data = dfmyd_SRER,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "SRER AQUA"), sh
  geom_line(data = dfmod_BONA,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "BONA TERA"), si
  geom_point(data = dfmod_BONA,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "BONA TERA"), sh
  geom_line(data = dfmyd_BONA,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "BONA AQUA"), si
  geom_point(data = dfmyd_BONA,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "BONA AQUA"), sh
  scale_color_manual(name = NULL, values=c("light blue", "pink", "blue", "red")) +
  labs(title = "Time Series", x = "Date", y = "NDVI")+
  theme(legend.position = "top",
        panel.background = element_rect(fill="white"),
        axis.line = element_line(color = "black"),
        axis.text.x = element_text(colour="black"),
        plot.title = element_text(hjust = 0.5))
```



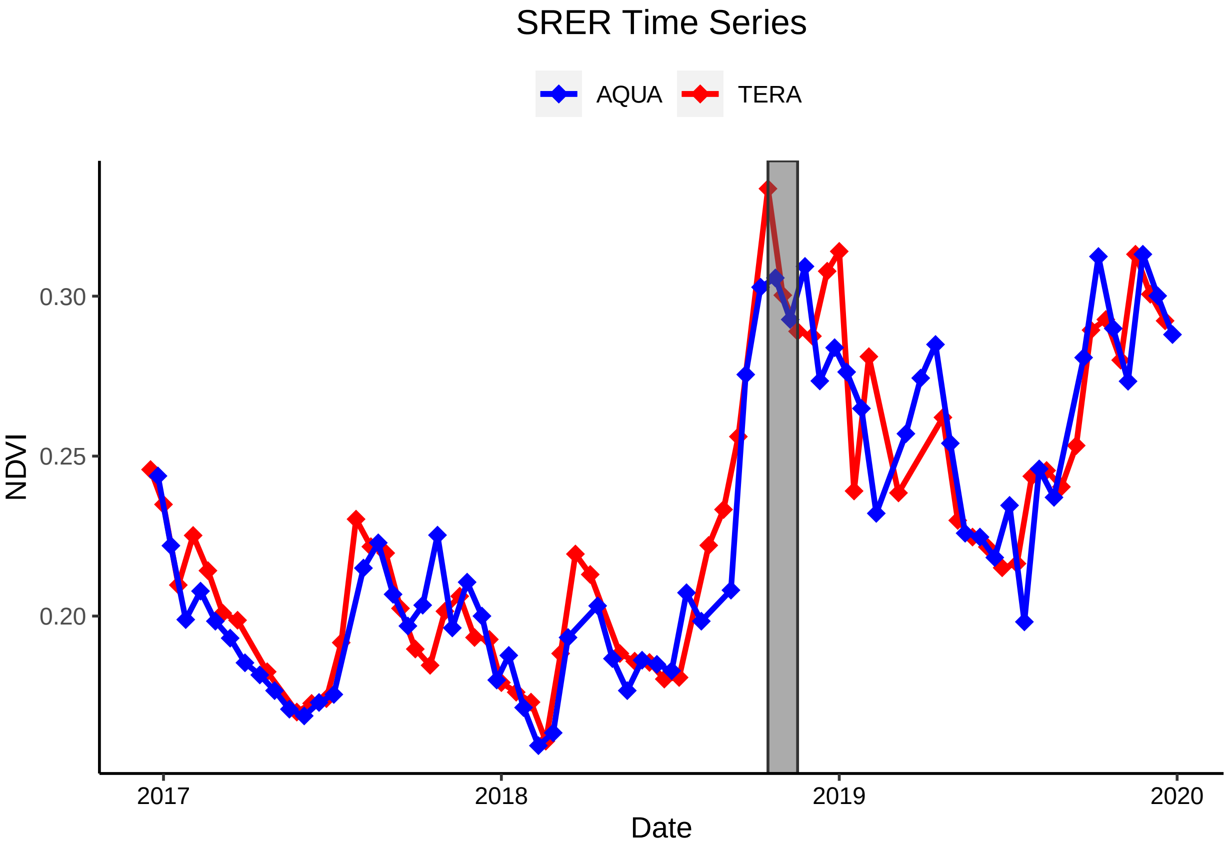
It makes sense that BONA has a higher NDVI than SRER, because BONA has much more vegetation and stays fairly green year-round.

5. Constrain a 3-week window for 'peak greenness' from MODIS and highlight it on your timeseries plot.

```
# I will use the TERA data to determine peak greenness for consistency
# TERA also has slightly higher estimates than AQUA on average

dfmod_SRER$Date[which(dfmod_SRER$MOD13Q1_006__250m_16_days_NDVI == max(dfmod_SRER$MOD13Q1_006__250m_16_
```

```
## [1] "2018-10-16"
```

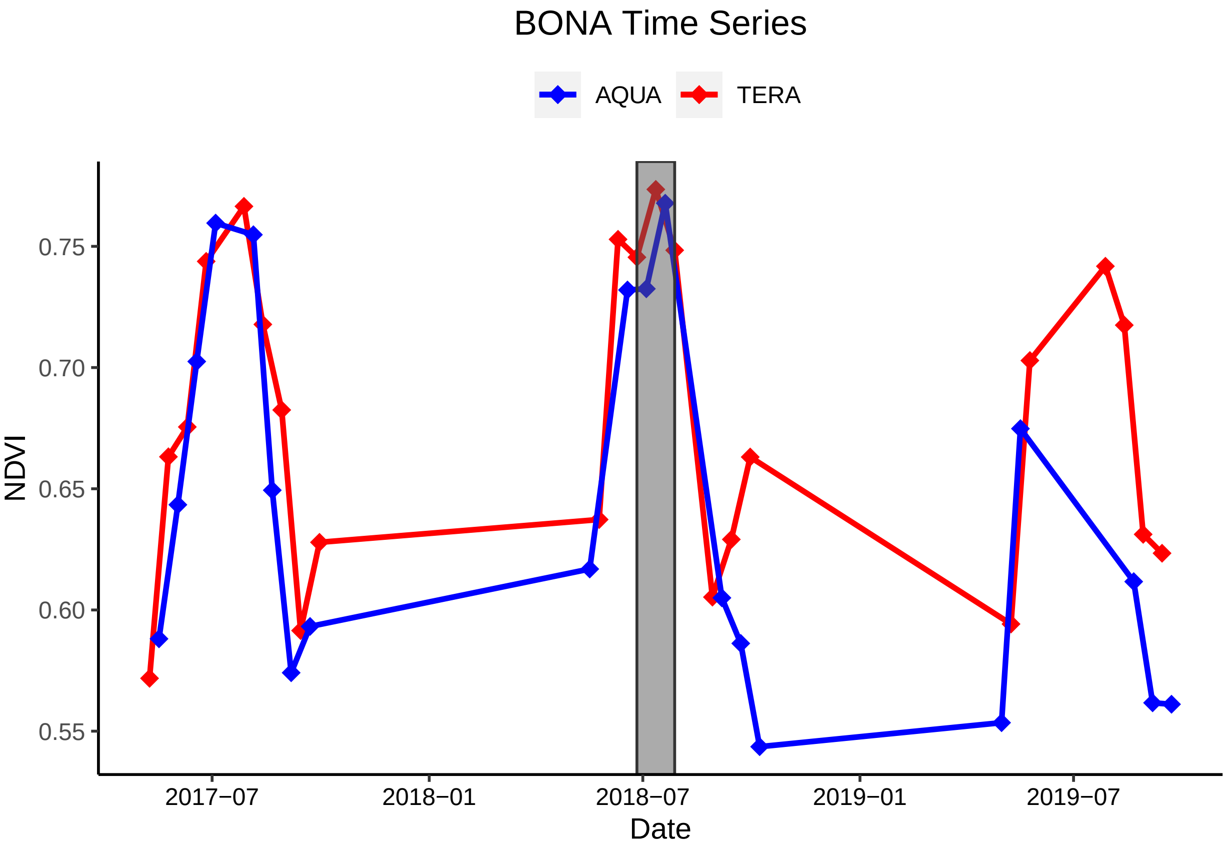
```
# SRER peak greenness, for about 3 weeks
rect <- data.frame(xmin=dfmod_SRER$Date[38], xmax=dfmod_SRER$Date[40], ymin=-Inf, ymax=Inf)

ggplot()+
  geom_line(data = dfmod_SRER,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA"), size=1)
  geom_point(data = dfmod_SRER,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA"), shape=
  geom_line(data = dfmyd_SRER,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"), size=1)
  geom_point(data = dfmyd_SRER,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"), shape=
  scale_color_manual(name = NULL, values=c("blue", "red")) +
  labs(title = "SRER Time Series", x = "Date", y = "NDVI")+
  geom_rect(data=rect, aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
            color="grey20",
            alpha=0.5,
            inherit.aes = FALSE) +
  theme(legend.position = "top",
        panel.background = element_rect(fill="white"),
        axis.line = element_line(color = "black"),
        axis.text.x = element_text(colour="black"),
        plot.title = element_text(hjust = 0.5))
```



```
dfmod_BONA$Date[which(dfmod_BONA$MOD13Q1_006__250m_16_days_NDVI == max(dfmod_BONA$MOD13Q1_006__250m_16_
```

```
## [1] "2018-07-12"
```

```
# BONA peak greenness, for about 3 weeks
rect <- data.frame(xmin=dfmod_BONA$Date[12], xmax=dfmod_BONA$Date[14], ymin=-Inf, ymax=Inf)

ggplot()+
  geom_line(data = dfmod_BONA,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA"), size=1)
  geom_point(data = dfmod_BONA,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA"), shape=
  geom_line(data = dfmyd_BONA,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"), size=1)
  geom_point(data = dfmyd_BONA,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"), shape=
  scale_color_manual(name = NULL, values=c("blue", "red")) +
  labs(title = "BONA Time Series", x = "Date", y = "NDVI")+
    geom_rect(data=rect, aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
              color="grey20",
              alpha=0.5,
              inherit.aes = FALSE) +
  theme(legend.position = "top",
        panel.background = element_rect(fill="white"),
        axis.line = element_line(color = "black"),
        axis.text.x = element_text(colour="black"),
        plot.title = element_text(hjust = 0.5))
```



6. Pull the canopy-level gcc90 from PhenoCam for the same site and the same time period as above.

Hint: Check the numbering of your PhenoCam on the PhenoCam gallery

```
library(phenocamapi)
```

```
## Warning: package 'phenocamapi' was built under R version 4.0.2

## Loading required package: data.table

## Warning: package 'data.table' was built under R version 4.0.2

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## Loading required package: rjson

## Warning: package 'rjson' was built under R version 4.0.2

##
## Attaching package: 'rjson'

## The following objects are masked from 'package:jsonlite':
##
##     fromJSON, toJSON

## Loading required package: RCurl

## Warning: package 'RCurl' was built under R version 4.0.2
```

```
##
## Attaching package: 'RCurl'

## The following object is masked from 'package:tidyr':
##
##     complete
```

```
library(xROI)
```

```
## Warning: package 'xROI' was built under R version 4.0.2
```

```
phenos=get_phenos()
head(phenos)
```

```
##                  site      lat        lon elev active utc_offset date_first
## 1: aafcottawacfiaf14e 45.29210  -75.76640   90   TRUE         -5 2020-04-27
## 2: aafcottawacfiaf14w 45.29210  -75.76640   90   TRUE         -5 2020-05-01
## 3:             acadia 44.37694  -68.26083  158   TRUE         -5 2007-03-15
## 4:      aguatibiaeast 33.62200 -116.86700 1086  FALSE         -8 2007-08-16
## 5:     aguatibianorth 33.60222 -117.34368 1090  FALSE         -8 2003-10-01
## 6:           ahwahnee 37.74670 -119.58160 1199   TRUE         -8 2008-08-28
##     date_last infrared                                      contact1
## 1: 2020-10-26        N Elizabeth Pattey <elizabeth.pattey@canada.ca>
## 2: 2020-10-26        N Elizabeth Pattey <elizabeth.pattey@canada.ca>
## 3: 2020-10-31        N                 Dee Morse <dee_morse@nps.gov>
## 4: 2019-01-25        N              Ann E Mebane <amebane@fs.fed.us>
## 5: 2006-10-25        N
## 6: 2020-10-31        N                 Dee Morse <dee_morse@nps.gov>
##                                     contact2
## 1: Luc Pelletier <luc.pelletier3@canada.ca>
## 2: Luc Pelletier <luc.pelletier3@canada.ca>
## 3:          John Gross <John_Gross@nps.gov>
## 4:   Kristi Savig <KSavig@air-resource.com>
## 5:
## 6:          John Gross <John_Gross@nps.gov>
##                                                 site_description site_type
## 1:  AAFC Site - Ottawa (On) - CFIA - Field F14 -East Flux Tower        II
## 2:  AAFC Site - Ottawa (On) - CFIA - Field F14 -West Flux Tower        II
## 3: Acadia National Park, McFarland Hill, near Bar Harbor, Maine       III
## 4:                            Agua Tibia Wilderness, California       III
## 5:                            Agua Tibia Wilderness, California       III
## 6:          Ahwahnee Meadow, Yosemite National Park, California       III
##                    group       camera_description camera_orientation flux_data
## 1:                  <NA> Campbell Scientific CCFC                 NE      TRUE
## 2:                  <NA> Campbell Scientific CCFC                WNW      TRUE
## 3: National Park Service                  unknown                 NE     FALSE
## 4:                  USFS                  unknown                 SW     FALSE
## 5:                  USFS                  unknown                 NE     FALSE
## 6: National Park Service                  unknown                  E     FALSE
##    flux_networks flux_sitenames
## 1:     <list[0]>           <NA>
## 2:     <list[1]>           <NA>
## 3:     <list[0]>
## 4:     <list[0]>
## 5:     <list[0]>
## 6:     <list[0]>
##                                             dominant_species primary_veg_type
## 1: Zea mays, Triticum aestivum, Brassica napus, Glycine max               AG
## 2: Zea mays, Triticum aestivum, Brassica napus, Glycine max               AG
## 3:                                                                        DB
## 4:                                                                        SH
## 5:                                                                        SH
## 6:                                                                        EN
##    secondary_veg_type site_meteorology MAT_site MAP_site MAT_daymet MAP_daymet
## 1:                 AG             TRUE      6.4      943       6.30        952
## 2:                 AG             TRUE      6.4      943       6.30        952
## 3:                 EN            FALSE       NA       NA       7.05       1439
## 4:                               FALSE       NA       NA      15.75        483
## 5:                               FALSE       NA       NA      16.00        489
## 6:                 GR            FALSE       NA       NA      12.25        871
##    MAT_worldclim MAP_worldclim koeppen_geiger ecoregion landcover_igbp
## 1:           6.0           863            Dfb         8             12
## 2:           6.0           863            Dfb         8             12
## 3:           6.5          1303            Dfb         8              5
## 4:          14.9           504            Csa        11              7
## 5:          13.8           729            Csa        11              7
## 6:          11.8           886            Csb         6              8
##    dataset_version1
## 1:               NA
## 2:               NA
## 3:               NA
## 4:               NA
## 5:               NA
## 6:               NA
##
## 1:  Camera funded by Agriculture and Agri-Food Canada (AAFC) Project J-001735 - Commercial inhibitor
## 2: Cameras funded by Agriculture and Agri-Food Canada (AAFC) Project J-001735 - Commercial inhibitors
## 3:
## 4:
## 5:
## 6:
##                            modified flux_networks_name flux_networks_url
## 1: 2020-05-04T10:46:30.065790-04:00               <NA>              <NA>
## 2: 2020-05-04T10:46:32.523976-04:00              OTHER
## 3: 2016-11-01T15:42:15.016778-04:00               <NA>              <NA>
## 4: 2016-11-01T15:42:15.086984-04:00               <NA>              <NA>
## 5: 2016-11-01T15:42:15.095277-04:00               <NA>              <NA>
## 6: 2016-11-01T15:42:15.111916-04:00               <NA>              <NA>
##    flux_networks_description
## 1:                      <NA>
## 2:       Other/Unaffiliated
## 3:                      <NA>
## 4:                      <NA>
## 5:                      <NA>
## 6:                      <NA>
```

```r
# obtaining the list of all the available ROI's on the PhenoCam server
rois <- get_rois()

# view what information is returned
```

```r
colnames(rois)
```

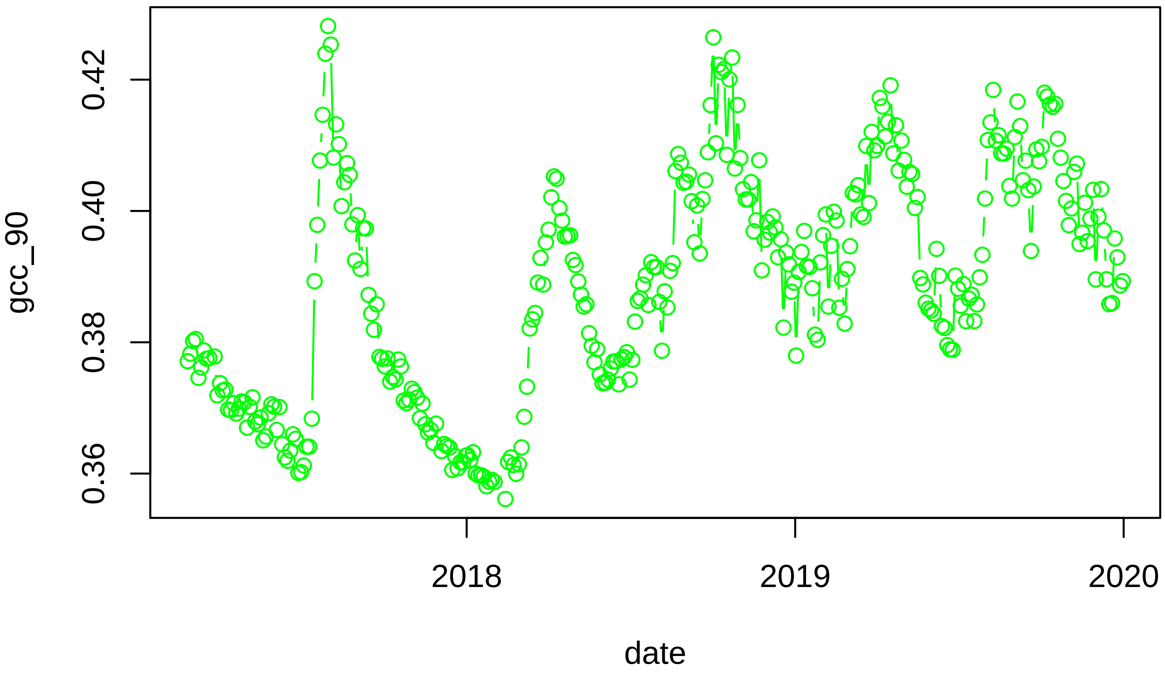
```
##  [1] "roi_name"          "site"              "lat"
##  [4] "lon"               "roitype"           "active"
##  [7] "show_link"         "show_data_link"    "sequence_number"
## [10] "description"       "first_date"        "last_date"
## [13] "site_years"        "missing_data_pct"  "roi_page"
## [16] "roi_stats_file"    "one_day_summary"   "three_day_summary"
## [19] "data_release"
```

```r
# to obtain the DB 1000 from dukehw
SRER_1000 <- get_pheno_ts(site = 'NEON.D14.SRER.DP1.00033', vegType = 'SH', roiID = 1000, type = '3day'
BONA_1000 <- get_pheno_ts(site = 'NEON.D19.BONA.DP1.00033', vegType = 'EN', roiID = 1000, type = '3day'

# what data are available
str(SRER_1000)
```

```
## Classes 'data.table' and 'data.frame':   450 obs. of  35 variables:
##  $ date                 : chr  "2017-02-25" "2017-02-28" "2017-03-03" "2017-03-06" ...
##  $ year                 : int  2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 ...
##  $ doy                  : int  56 59 62 65 68 71 74 77 80 83 ...
##  $ image_count          : int  98 114 120 120 120 123 123 123 41 0 ...
##  $ midday_filename      : chr  "NEON.D14.SRER.DP1.00033_2017_02_25_120005.jpg" "NEON.D14.SRER.DP1.0003
##  $ midday_r             : num  74.6 67.7 79.5 73.9 74.5 ...
##  $ midday_g             : num  63.4 57.1 67.4 62.9 62.9 ...
##  $ midday_b             : num  32.2 27 30.9 31.3 30.7 ...
##  $ midday_gcc           : num  0.373 0.376 0.379 0.374 0.374 ...
##  $ midday_rcc           : num  0.438 0.446 0.447 0.44 0.443 ...
##  $ r_mean               : num  75 80.5 80.9 80.8 83.6 ...
##  $ r_std                : num  15 17.3 18.3 18.9 22.4 ...
##  $ g_mean               : num  64.9 69.8 70.2 70.2 72.8 ...
##  $ g_std                : num  13.8 15.8 17.2 17.9 21.2 ...
##  $ b_mean               : num  34.7 37.1 37.2 37.2 40.3 ...
##  $ b_std                : num  9.79 11.35 12.94 13.44 15.32 ...
##  $ gcc_mean             : num  0.372 0.373 0.373 0.374 0.371 ...
##  $ gcc_std              : num  0.00391 0.00435 0.00513 0.00521 0.00295 0.00287 0.00397 0.00359 0.00298
##  $ gcc_50               : num  0.372 0.373 0.373 0.373 0.371 ...
##  $ gcc_75               : num  0.374 0.377 0.378 0.379 0.373 ...
##  $ gcc_90               : num  0.377 0.378 0.38 0.381 0.375 ...
##  $ rcc_mean             : num  0.431 0.431 0.432 0.432 0.428 ...
##  $ rcc_std              : num  0.0129 0.016 0.0143 0.0145 0.0133 ...
##  $ rcc_50               : num  0.432 0.434 0.434 0.434 0.429 ...
##  $ rcc_75               : num  0.441 0.442 0.443 0.444 0.44 ...
##  $ rcc_90               : num  0.443 0.449 0.45 0.45 0.447 ...
##  $ max_solar_elev       : num  49.7 50.8 51.9 53.1 54.3 ...
##  $ snow_flag            : logi  NA NA NA NA NA NA ...
##  $ outlierflag_gcc_mean: logi  NA NA NA NA NA NA ...
##  $ outlierflag_gcc_50  : logi  NA NA NA NA NA NA ...
##  $ outlierflag_gcc_75  : logi  NA NA NA NA NA NA ...
##  $ outlierflag_gcc_90  : logi  NA NA NA NA NA NA ...
##  $ YEAR                 : int  2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 ...
##  $ DOY                  : int  56 59 62 65 68 71 74 77 80 83 ...
##  $ YYYYMMDD             : chr  "2017-02-25" "2017-02-28" "2017-03-03" "2017-03-06" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

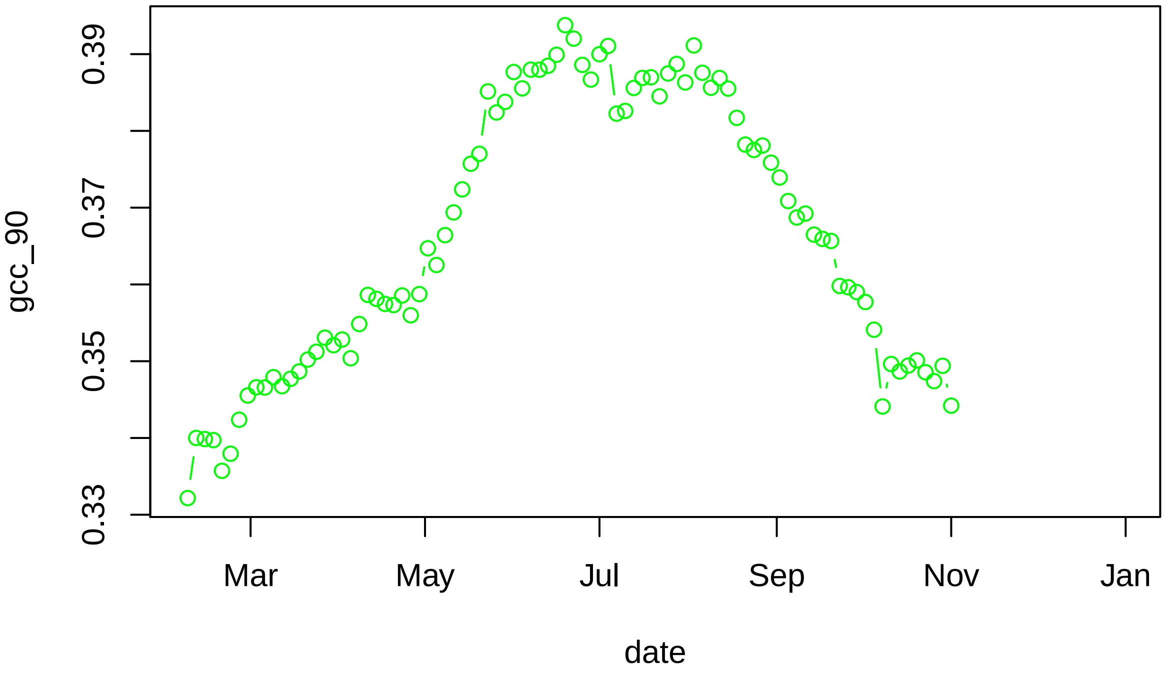
I just want to take a look at these timeseries before I plot them together:

```
# Make dates match NDVI data
SRER_1000 <- SRER_1000 %>%
  filter(year != 2020)

# date variable into date format
SRER_1000[,date:=as.Date(date)]

# plot gcc_90
SRER_1000[,plot(date, gcc_90, col = 'green', type = 'b')]
```



```
## NULL
```

```
# Make dates match NDVI data
BONA_1000 <- BONA_1000 %>%
  filter(year != 2020)

# date variable into date format
BONA_1000[,date:=as.Date(date)]

# plot gcc_90
BONA_1000[,plot(date, gcc_90, col = 'green', type = 'b')]
```

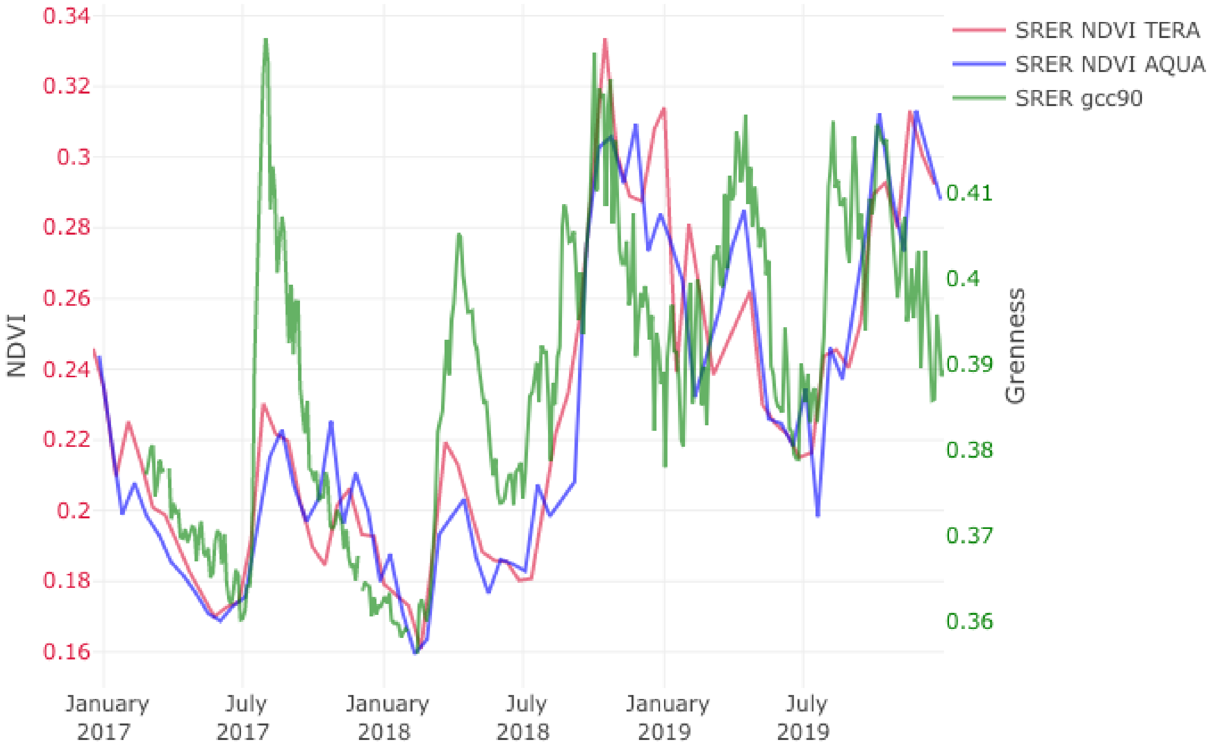


```
## NULL
```

The BONA phenocam doesn't have 2018 data, so I'll plot the peak greenness overlap in 2019 instead.

7. Plot the PhenoCam and MODIS timeseries on the same plot.

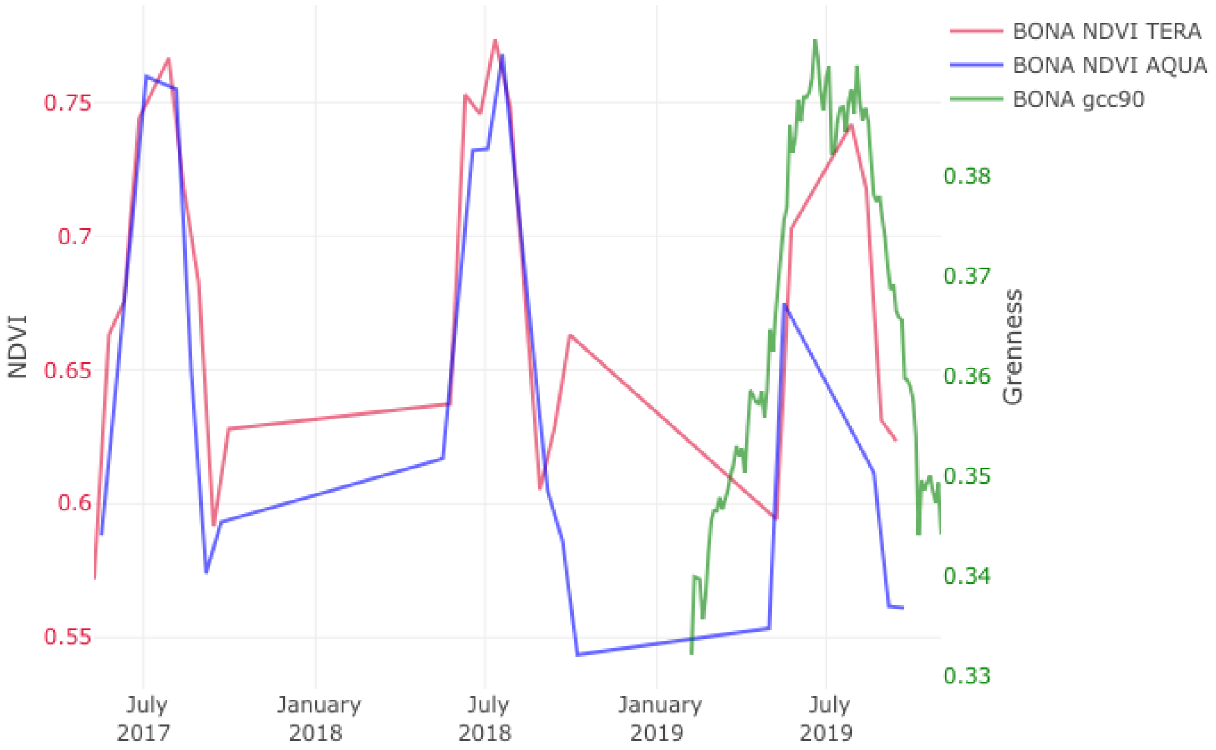
```
ay <- list(
  tickfont = list(color = "green"),
  overlaying = "y",
  side = "right",
  title = "Grenness",
  side='right',
  zeroline = FALSE,
  showline = FALSE,
  showgrid= FALSE,
  title='SRER'
)


fig <- plot_ly()
fig <- fig %>% add_lines(x = dfmod_SRER$Date, y = dfmod_SRER$MOD13Q1_006__250m_16_days_NDVI, name = "SRI
fig <- fig %>% add_lines(x = dfmyd_SRER$Date, y =dfmyd_SRER$MYD13Q1_006__250m_16_days_NDVI, name = "SREI
fig <- fig %>% add_lines(x = SRER_1000$date, y =SRER_1000$gcc_90, name = "SRER gcc90", yaxis = "y2", lin
fig <- fig %>% layout(
     yaxis2 = ay,
     yaxis= list(tickfont = list(color = "crimson"), title = "NDVI"),
    xaxis = list(type = 'date',
        tickformat = " %B<br>%Y")#,
        #legend=list(x=500, y=.5))
  )

fig <- fig %>% layout(legend = list(x = 1, y = 1))
fig
```



This plot makes it more clear that sometimes SRER experiences peak greenness in fall (but not in 2017). Looking at climate data might give us a clue as to why this is.

```
ay <- list(
  tickfont = list(color = "green"),
  overlaying = "y",
  side = "right",
  title = "Grenness",
  side='right',
  zeroline = FALSE,
  showline = FALSE,
  showgrid= FALSE,
  title='BONA'
)


fig <- plot_ly()
fig <- fig %>% add_lines(x = dfmod_BONA$Date, y = dfmod_BONA$MOD13Q1_006__250m_16_days_NDVI, name = "BO
fig <- fig %>% add_lines(x = dfmyd_BONA$Date, y =dfmyd_BONA$MYD13Q1_006__250m_16_days_NDVI, name = "BONA
fig <- fig %>% add_lines(x = BONA_1000$date, y =BONA_1000$gcc_90, name = "BONA gcc90", yaxis = "y2", li
fig <- fig %>% layout(
     yaxis2 = ay,
     yaxis= list(tickfont = list(color = "crimson"), title = "NDVI"),
    xaxis = list(type = 'date',
        tickformat = " %B<br>%Y")#,
        #legend=list(x=500, y=.5))
  )
```

```
fig <- fig %>% layout(legend = list(x = 1, y = 1))
fig
```



BONA phenocam greenness matches up with BONA NDVI pretty well in 2019 at least.

8. Constrain a 3-week window for 'peak greeness' from PhenoCam and highlight it on your timeseries. *Hint: Remember our PhenoCam discussions regarding early 'extra green' leaves? You'll need to use some logic for this, not just a `max`*

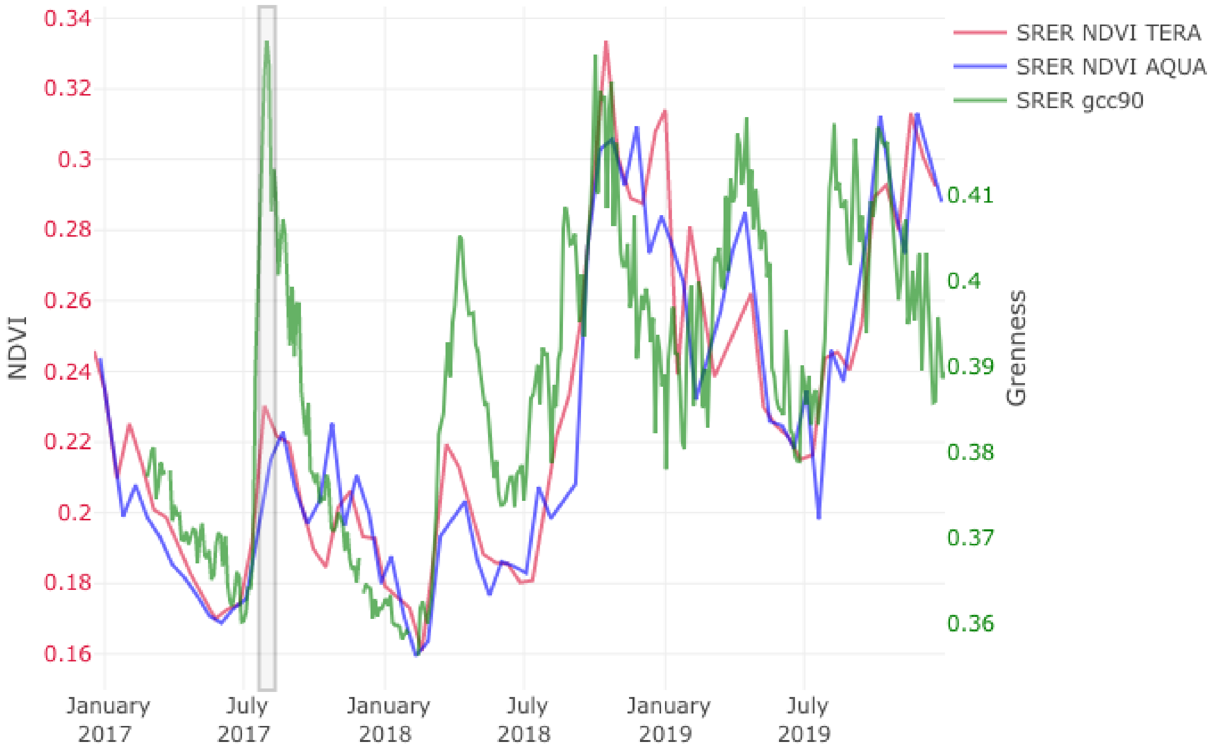
The green from spring leaf-out does not exceed the later season greenness at SRER and BONA, so we shouldn't have a problem using max.

```
SRER_1000$date[which(SRER_1000$gcc_90 == max(SRER_1000$gcc_90, na.rm = T))]
```

```
## [1] "2017-07-31"
```

```
ay <- list(
  tickfont = list(color = "green"),
  overlaying = "y",
  side = "right",
  title = "Grenness",
  side='right',
  zeroline = FALSE,
  showline = FALSE,
  showgrid= FALSE,
  title='SRER'
)
```

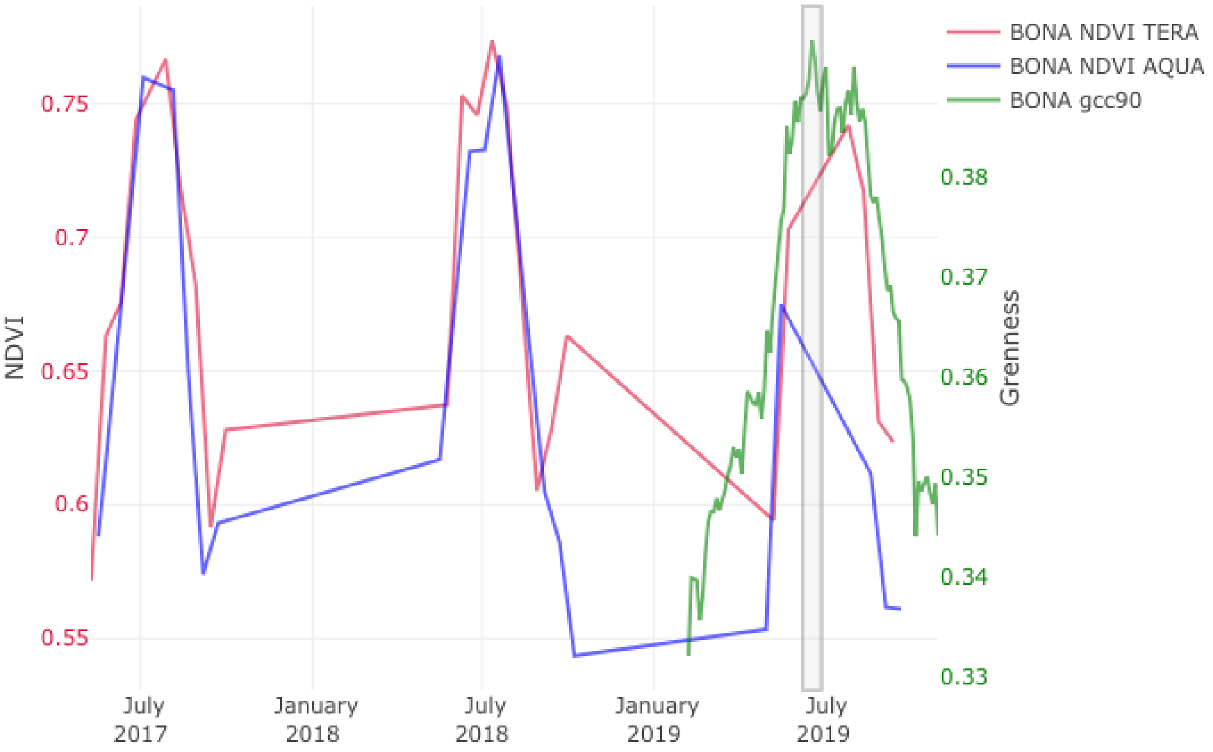
```
fig <- plot_ly()
fig <- fig %>% add_lines(x = dfmod_SRER$Date, y = dfmod_SRER$MOD13Q1_006__250m_16_days_NDVI, name = "SR
fig <- fig %>% add_lines(x = dfmyd_SRER$Date, y =dfmyd_SRER$MYD13Q1_006__250m_16_days_NDVI, name = "SRE
fig <- fig %>% add_lines(x = SRER_1000$date, y =SRER_1000$gcc_90, name = "SRER gcc90", yaxis = "y2", li
fig <- fig %>% layout(
     yaxis2 = ay,
     yaxis= list(tickfont = list(color = "crimson"), title = "NDVI"),
    xaxis = list(type = 'date',
                 tickformat = " %B<br>%Y"),
    shapes =list(type = 'rect',
            # x-reference is assigned to the x-values
            xref = 'x',
            # y-reference is assigned to the plot paper [0,1]
            yref= 'paper',
            x0 = '2017-07-21',
            y0= 0,
            x1 = '2017-08-11',
            y1 = 1,
            fillcolor = '#d3d3d3',
            opacity = 0.2)
        )

fig <- fig %>% layout(legend = list(x = 1, y = 1))
fig
```



```
BONA_1000$date[which(BONA_1000$gcc_90 == max(BONA_1000$gcc_90, na.rm = T))]
```

```
## [1] "2019-06-19"
```

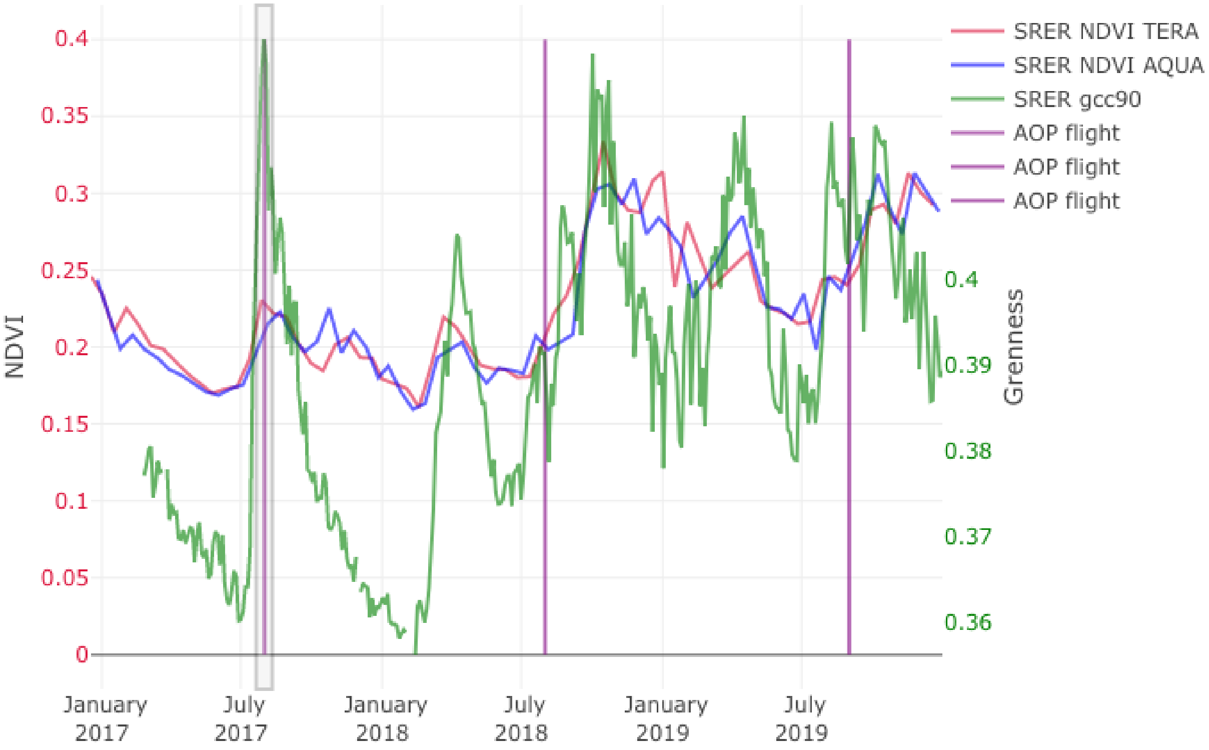
```
ay <- list(
  tickfont = list(color = "green"),
  overlaying = "y",
  side = "right",
  title = "Grenness",
  side='right',
  zeroline = FALSE,
  showline = FALSE,
  showgrid= FALSE,
  title='BONA'
)


fig <- plot_ly()
fig <- fig %>% add_lines(x = dfmod_BONA$Date, y = dfmod_BONA$MOD13Q1_006__250m_16_days_NDVI, name = "BO
fig <- fig %>% add_lines(x = dfmyd_BONA$Date, y =dfmyd_BONA$MYD13Q1_006__250m_16_days_NDVI, name = "BONA
fig <- fig %>% add_lines(x = BONA_1000$date, y =BONA_1000$gcc_90, name = "BONA gcc90", yaxis = "y2", li
fig <- fig %>% layout(
     yaxis2 = ay,
     yaxis= list(tickfont = list(color = "crimson"), title = "NDVI"),
    xaxis = list(type = 'date',
        tickformat = " %B<br>%Y"),
    shapes =list(type = 'rect',
            # x-reference is assigned to the x-values
            xref = 'x',
            # y-reference is assigned to the plot paper [0,1]
            yref= 'paper',
            x0 = '2019-06-09',
            y0= 0,
            x1 = '2019-06-29',
            y1 = 1,
            fillcolor = '#d3d3d3',
            opacity = 0.2)
  )

fig <- fig %>% layout(legend = list(x = 1, y = 1))
fig
```



9. Find the timing of the AOP flights that have occurred at your sites over the same time period. Add those dates as a vertical line.

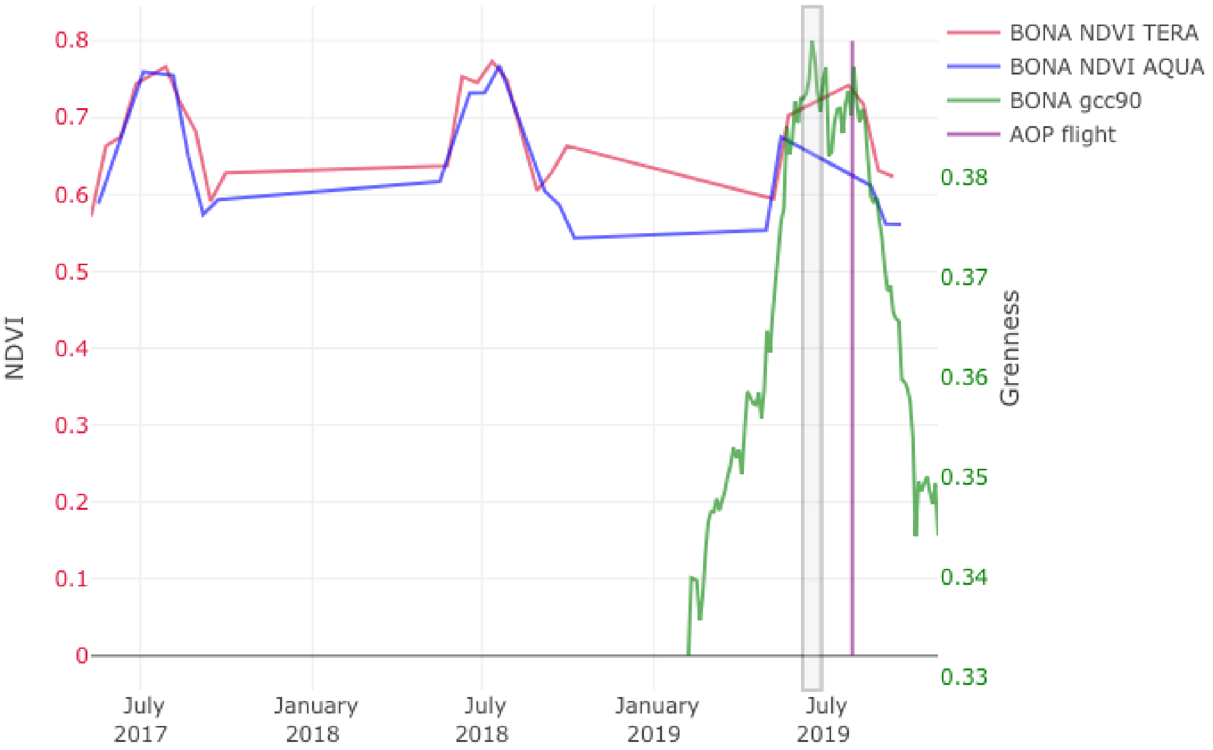
SRER AOP flights: 2017-08, 2018-08, 2019-09 BONA AOP flights: 2019-08

```
ay <- list(
  tickfont = list(color = "green"),
  overlaying = "y",
  side = "right",
  title = "Grenness",
  side='right',
  zeroline = FALSE,
  showline = FALSE,
  showgrid= FALSE,
  title='SRER'
)


fig <- plot_ly()
fig <- fig %>% add_lines(x = dfmod_SRER$Date, y = dfmod_SRER$MOD13Q1_006__250m_16_days_NDVI, name = "SRI
fig <- fig %>% add_lines(x = dfmyd_SRER$Date, y =dfmyd_SRER$MYD13Q1_006__250m_16_days_NDVI, name = "SREI
fig <- fig %>% add_lines(x = SRER_1000$date, y =SRER_1000$gcc_90, name = "SRER gcc90", yaxis = "y2", lin
fig <- fig %>% layout(
     yaxis2 = ay,
     yaxis= list(tickfont = list(color = "crimson"), title = "NDVI"),
    xaxis = list(type = 'date',
                 tickformat = " %B<br>%Y"),
    shapes =list(type = 'rect',
            # x-reference is assigned to the x-values
            xref = 'x',
            # y-reference is assigned to the plot paper [0,1]
            yref= 'paper',
            x0 = '2017-07-21',
            y0= 0,
            x1 = '2017-08-11',
            y1 = 1,
            fillcolor = '#d3d3d3',
            opacity = 0.2)
        )

fig <- fig %>% layout(legend = list(x = 1, y = 1))
fig <- fig %>% add_segments(x ='2017-08', xend = '2017-08', y = 0, yend = 0.4, name = "AOP flight", yax
fig
```



```
ay <- list(
  tickfont = list(color = "green"),
  overlaying = "y",
  side = "right",
  title = "Grenness",
  side='right',
  zeroline = FALSE,
  showline = FALSE,
  showgrid= FALSE,
  title='BONA'
)
```

```
fig <- plot_ly()
fig <- fig %>% add_lines(x = dfmod_BONA$Date, y = dfmod_BONA$MOD13Q1_006__250m_16_days_NDVI, name = "BO
fig <- fig %>% add_lines(x = dfmyd_BONA$Date, y =dfmyd_BONA$MYD13Q1_006__250m_16_days_NDVI, name = "BONA
fig <- fig %>% add_lines(x = BONA_1000$date, y =BONA_1000$gcc_90, name = "BONA gcc90", yaxis = "y2", li
fig <- fig %>% layout(
     yaxis2 = ay,
     yaxis= list(tickfont = list(color = "crimson"), title = "NDVI"),
    xaxis = list(type = 'date',
        tickformat = " %B<br>%Y"),
    shapes =list(type = 'rect',
            # x-reference is assigned to the x-values
            xref = 'x',
            # y-reference is assigned to the plot paper [0,1]
            yref= 'paper',
            x0 = '2019-06-09',
            y0= 0,
            x1 = '2019-06-29',
            y1 = 1,
            fillcolor = '#d3d3d3',
            opacity = 0.2)
  )

fig <- fig %>% layout(legend = list(x = 1, y = 1))
fig <- fig %>% add_segments(x ='2019-08', xend = '2019-08', y = 0, yend = 0.8, name = "AOP flight", yax
fig
```



It seems that the AOP flights sometimes make and sometimes miss actual peak greenness.