INF 550 Section 7.4.5

Natasha Wesely

2022-11-02

```
# Load necessary packages into R
library(getPass)            # A micro-package for reading passwords
library(httr)               # To send a request to the server/receive a response from the server
library(jsonlite)           # Implements a bidirectional mapping between JSON data and the most importa
library(ggplot2)            # Functions for graphing and mapping
library(tidyr)              # Function for working with tabular data
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
```

Question 1

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

I am choosing the NEON sites JORN & TEAK.

Question 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.

```r
# Set Up the Output Directory
# Set your input directory, and create an output directory for the results.
outDir <- file.path('./data/')                  # Create an output directory if it doesn't exist
suppressWarnings(dir.create(outDir))
```

```r
# Login to Earth Data
# To submit a request, you must first login to the AppEEARS API. Use your private R Script to enter you
load('EARTHDATA_Token2.Rdata')
exists('EARTHDATA_User')
```

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

```r
# rename my user name id for simplicity
user = EARTHDATA_User
password = EARTHDATA_Password

# Decode the username and password to be used to post login request.
secret <- jsonlite::base64_enc(paste(user, password, sep = ":"))  # Encode the string of username and p
# Next, assign the AppEEARS API URL to a static variable.
API_URL = "https://appeears.earthdatacloud.nasa.gov/api/"  # Set the AppEEARS API to a variable
```

```r
# Use the httr package to post your username and password. A successful login will provide you with a t

# 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)
```

```
## {
##     "token_type": "Bearer",
##     "token": "kYedmm4e4iZHwQuUpUPpli-5OXalYqvCpV_5foc-BDG6as4IyUixS4AqtEmjbXafTws5wtc7BdJLjGYsTln2Hw
##     "expiration": "2022-11-04T21:53:04Z"
## }
##
```

```r
# Query Available Products
# The product API provides details about all of the products and layers available in AppEEARS. For more
#
# Below, call the product API to list all of the products available in AppEEARS.

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)
```

```r
# Search and Explore Available Products
# Create a list indexed by product name to make it easier to query a specific product.

# 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
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 163 products."
```

```r
# Using the info above, Create a list of desired products.

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

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

```r
# Search and Explore Available Layers
# This API call will list all of the layers available for a given product. Each product is referenced by
# First, request the layers for the MOD13Q1.006 product.

# Request layers for the 1st product in the list: MOD13Q1.006
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))
```

```
##  [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"
```

```r
# Request layers for the 1st product in the list: MYD13Q1.006
MYD13Q1.006_req <- GET(paste0(API_URL,"product/", desired_products[2]))  # Request the info of a produc
MYD13Q1.006_content <- content(MYD13Q1.006_req)                             # Retrieve content of the r
MYD13Q1.006_response <- toJSON(MYD13Q1.006_content, auto_unbox = TRUE)      # Convert the content to JS
remove(MYD13Q1.006_req, MYD13Q1.006_content)                                # Remove the variables that
#prettify(MOD13Q1_response)                                               # Print the prettified response
names(fromJSON(MYD13Q1.006_response))
```

```
##  [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"
```

```
# lastly, select the desired layers and pertinent products and make a data frame using this information

# Create a vector of desired layers
desired_layers <- c("_250m_16_days_NDVI")

# Create a data frame including the desired data products and layers
layers <- data.frame(product = desired_products, layer = desired_layers)
```

```
# the Submit task API call provides a way to submit a new request to be processed. It can accept data v

# in this section, begin by setting up the information needed for a nested data frame that will be late

# For point requests, beside the date range and desired layers information, the coordinates property mu

# We'll start by requesting point-based data for NEON.D17.SOAP and NEON.D17.SJER:

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 <- [2000,2016]       # If recurring = True, set yearRange, change start/end date to MM-DD

lat <- c(32.59069, 37.00583)        # Latitude of the point sites
lon <- c(-106.84254, -119.00602)    # Longitude of the point sites
# id <- c("0","1")                      # ID for the point sites
id = "0"
category <- c("JORN", "TEAK") # Category for point sites

taskName <- 'NEON JORN TEAK NDVI'           # Enter name of the task here
taskType <- 'point'                    # Specify the task type, it can be either "area" or "point"
```

```
# to successfully submit a task, the JSON object should be structured in a certain way. The code chunk

# 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
```

```r
task <- list(task_info, taskName, taskType)                # Create a nested list
names(task) <- c("params", "task_name", "task_type")       # Assign names
remove(date, coordinates, task_info)                # Remove the variables that are not needed anymore
```

```r
# toJSON function from jsonlite package converts the type of data frame to a string that can be recogni

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

```r
# Submit a Task Request
# Token information is needed to submit a request. Below the login token is assigned to a variable.

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

# Below, post a call to the API task service, using the task_json created above

# 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
```

```
## {
##     "task_id": "e2ae37cd-ce77-45b7-bca2-aa928a0298b5",
##     "status": "pending"
## }
##
```

```r
# Retrieve Task Status
# This API call will list all of the requests associated with your user account, automatically sorted b

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)
```

```
## [
##     {
##         "params": {
##             "dates": [
##                 {
##                     "endDate": "12-31-2019",
##                     "startDate": "01-01-2017"
##                 }
##             ],
```

```
##             "layers": [
##                 {
##                     "layer": "_250m_16_days_NDVI",
##                     "product": "MOD13Q1.006"
##                 },
##                 {
##                     "layer": "_250m_16_days_NDVI",
##                     "product": "MYD13Q1.006"
##                 }
##             ]
##         },
##         "status": "pending",
##         "created": "2022-11-02T21:54:32.835890",
##         "task_id": "e2ae37cd-ce77-45b7-bca2-aa928a0298b5",
##         "updated": "2022-11-02T21:54:33.049114",
##         "user_id": "nkw54@nau.edu",
##         "estimate": {
##             "request_size": 272
##         },
##         "has_swath": false,
##         "task_name": "NEON JORN TEAK NDVI",
##         "task_type": "point",
##         "api_version": "v1",
##         "svc_version": "3.16",
##         "web_version": {
##
##         },
##         "has_nsidc_daac": false,
##         "expires_on": "2022-12-02T21:54:33.049114"
##     },
##     {
##         "error": {
##
##         },
##         "params": {
##             "dates": [
##                 {
##                     "endDate": "12-31-2019",
##                     "startDate": "01-01-2017"
##                 }
##             ],
##             "layers": [
##                 {
##                     "layer": "_250m_16_days_NDVI",
##                     "product": "MOD13Q1.006"
##                 },
##                 {
##                     "layer": "_250m_16_days_NDVI",
##                     "product": "MYD13Q1.006"
##                 }
##             ]
##         },
##         "status": "done",
##         "created": "2022-11-02T21:53:05.503623",
```

```
##         "task_id": "f3591b57-9246-43e0-8ac1-f970e64a5dc6",
##         "updated": "2022-11-02T21:54:31.806921",
##         "user_id": "nkw54@nau.edu",
##         "attempts": 1,
##         "estimate": {
##             "request_size": 272
##         },
##         "retry_at": {
##
##         },
##         "completed": "2022-11-02T21:54:31.796739",
##         "has_swath": false,
##         "task_name": "NEON JORN TEAK NDVI",
##         "task_type": "point",
##         "api_version": "v1",
##         "svc_version": "3.16",
##         "web_version": {
##
##         },
##         "size_category": "0",
##         "has_nsidc_daac": false,
##         "expires_on": "2022-12-02T21:54:31.806921"
##     }
## ]
##
```

```
# task_id that was generated when submitting your request can also be used to retrieve a task status.

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
```

```
## {
##     "params": {
##         "dates": [
##             {
##                 "endDate": "12-31-2019",
##                 "startDate": "01-01-2017"
##             }
##         ],
##         "layers": [
##             {
##                 "layer": "_250m_16_days_NDVI",
##                 "product": "MOD13Q1.006"
##             },
##             {
##                 "layer": "_250m_16_days_NDVI",
##                 "product": "MYD13Q1.006"
##             }
```

```
##         ],
##         "coordinates": [
##             {
##                 "id": "0",
##                 "category": "JORN",
##                 "latitude": 32.5907,
##                 "longitude": -106.8425
##             },
##             {
##                 "id": "0",
##                 "category": "TEAK",
##                 "latitude": 37.0058,
##                 "longitude": -119.006
##             }
##         ]
##     },
##     "status": "pending",
##     "created": "2022-11-02T21:54:32.835890",
##     "task_id": "e2ae37cd-ce77-45b7-bca2-aa928a0298b5",
##     "updated": "2022-11-02T21:54:33.049114",
##     "user_id": "nkw54@nau.edu",
##     "estimate": {
##         "request_size": 272
##     },
##     "has_swath": false,
##     "task_name": "NEON JORN TEAK NDVI",
##     "task_type": "point",
##     "api_version": "v1",
##     "svc_version": "3.16",
##     "web_version": {
##
##     },
##     "has_nsidc_daac": false,
##     "expires_on": "2022-12-02T21:54:33.049114"
## }
##
```

```
# Retrieve the task status every 5 seconds. The task status should be done to be able to download the o

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)
}
```

```
## [1] "pending"
## [1] "pending"
## [1] "pending"
## [1] "pending"
## [1] "pending"
## [1] "pending"
```

```
## [1] "pending"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "processing"
## [1] "done"
```

```
# Download a Request
# Explore Files in Request Output
# Before downloading the request output, examine the files contained in the request output.

# 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)
```


```
## {
##     "files": [
##         {
##             "sha256": "74758179a2bc2ee4d0ef8a0380d48935b17e8a88761f14517b2abe9152490c08",
##             "file_id": "db1e417e-8d77-4464-91d9-2b7eaa23f629",
##             "file_name": "NEON-JORN-TEAK-NDVI-MOD13Q1-006-results.csv",
##             "file_size": 33796,
##             "file_type": "csv"
##         },
##         {
##             "sha256": "fbb0f0fbc01df5195fe8face7a35d5535e31a48f0876f684d4b853850ec7fcda",
##             "file_id": "3eeb8047-c5a3-4ace-ba58-295170bd6799",
##             "file_name": "NEON-JORN-TEAK-NDVI-MYD13Q1-006-results.csv",
##             "file_size": 33966,
##             "file_type": "csv"
##         },
##         {
##             "sha256": "415070cffd6eace982f2597bfb258bd0c961d40c273aa28410d93501af859824",
##             "file_id": "fe2437f5-7790-4d2b-a3b7-963c19ea924f",
##             "file_name": "NEON-JORN-TEAK-NDVI-granule-list.txt",
##             "file_size": 15260,
##             "file_type": "txt"
##         },
##         {
##             "sha256": "80d94ec7d7c5c5edfd05b1882fdf95053137a555802c15cde356523c332d6725",
##             "file_id": "405491bd-b17b-42a8-9366-434d5086cbc8",
##             "file_name": "NEON-JORN-TEAK-NDVI-request.json",
```

```
##             "file_size": 882,
##             "file_type": "json"
##         },
##         {
##             "sha256": "535292525e04ac5b8a7f44d45219619624119b2cbda77cab634a658f65b99a5b",
##             "file_id": "7f80c17d-d471-4ddd-b3fe-1b78298a8030",
##             "file_name": "NEON-JORN-TEAK-NDVI-MOD13Q1-006-metadata.xml",
##             "file_size": 17159,
##             "file_type": "xml"
##         },
##         {
##             "sha256": "536e6e28640aaa02ab940b5f4ece0f49ea30c30bcb86c39ee9f0e01cb92d4283",
##             "file_id": "1f166a76-231c-4e96-9aea-8b4087c066c0",
##             "file_name": "NEON-JORN-TEAK-NDVI-MYD13Q1-006-metadata.xml",
##             "file_size": 17158,
##             "file_type": "xml"
##         },
##         {
##             "sha256": "232f55899a2c0249541734624c62122b85128cde4bbac59ab53c3bfc05e0544f",
##             "file_id": "0002fd6f-4366-4cf6-be73-bd90340b4643",
##             "file_name": "README.md",
##             "file_size": 17311,
##             "file_type": "txt"
##         }
##     ],
##     "created": "2022-11-02T21:55:10.972186",
##     "task_id": "e2ae37cd-ce77-45b7-bca2-aa928a0298b5",
##     "updated": "2022-11-02T21:56:13.500453",
##     "bundle_type": "point"
## }
##
```

```
# Download Files in a Request (Automation)
# The bundle API provides information about completed tasks. For any completed task, a bundle can be qu

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))
}
```

```
##   |                                                                              |
##   |                                                                              |
##   |                                                                              |
##   |                                                                              |
##   |                                                                              |
```

```
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
##   |                                                                      |
```

Question 3

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

```
# Explore AppEEARS Quality Service
# The quality API provides quality details about all of the data products available in AppEEARS. Below
#
# First, reset pagination to include offset which allows you to set the number of results to skip befor

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
```


```
## [
##     {
##         "ProductAndVersion": "HLSS30.020",
##         "Layer": "B10",
##         "QualityProductAndVersion": "HLSS30.020",
##         "QualityLayers": [
##             "Fmask"
##         ],
##         "Continuous": false,
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "HLSS30.020",
##         "Layer": "B11",
##         "QualityProductAndVersion": "HLSS30.020",
##         "QualityLayers": [
##             "Fmask"
##         ],
##         "Continuous": false,
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "HLSS30.020",
##         "Layer": "B12",
##         "QualityProductAndVersion": "HLSS30.020",
##         "QualityLayers": [
##             "Fmask"
```

```
##         ],
##         "Continuous": false,
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "ASTGTM_NC.003",
##         "Layer": "ASTER_GDEM_DEM",
##         "QualityProductAndVersion": "ASTGTM_NUMNC.003",
##         "QualityLayers": [
##             "ASTER_GDEM_NUM"
##         ],
##         "Continuous": false,
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "CU_LC08.001",
##         "Layer": "SRB1",
##         "QualityProductAndVersion": "CU_LC08.001",
##         "QualityLayers": [
##             "PIXELQA"
##         ],
##         "Continuous": false,
##         "VisibleToWorker": true
##     },
##     {
##         "ProductAndVersion": "CU_LC08.001",
##         "Layer": "SRB2",
##         "QualityProductAndVersion": "CU_LC08.001",
##         "QualityLayers": [
##             "PIXELQA"
##         ],
##         "Continuous": false,
##         "VisibleToWorker": true
##     }
## ]
##
```

```
# List Quality Layers
# This API call will list all of the quality layer information for a product. For more information visi

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
##     }
## ]
##
```

```
# Inspect Quality Values
# This API call will list all of the values for a given quality layer.
```

```r
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": {
##
##         }
##     }
## ]
##
```

```
# Decode Quality Values
# This API call will decode the bits for a given quality value.

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"
##     }
## }
##
```

```
# Load Request Output and Visualize
# Here, load the CSV file containing the results from your request using readr package, and create some
#
# Load a CSV
# Use the readr package to load the CSV file containing the results from the AppEEARS request.

# Make a list of csv files in the output directory
files <- list.files(outDir, pattern = "\\NEON-JORN-TEAK-NDVI-MOD13Q1-006-results.csv$")
# Read the results
```

```
dfMOD <- read_csv(paste0(outDir,"/", files))
```

```
## Rows: 140 Columns: 29
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (21): Category, MODIS_Tile, MOD13Q1_006__250m_16_days_VI_Quality_bitmas...
## dbl   (7): ID, Latitude, Longitude, MOD13Q1_006_Line_Y_250m, MOD13Q1_006_Sam...
## date  (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# filter for bad data
dfMOD = dfMOD[dfMOD$MOD13Q1_006__250m_16_days_VI_Quality_MODLAND_Description == "VI produced with good c

# Make a list of csv files in the output directory
files <- list.files(outDir, pattern = "\\NEON-JORN-TEAK-NDVI-MYD13Q1-006-results.csv$")
# Read the results
dfMYD <- read_csv(paste0(outDir,"/", files))
```

```
## Rows: 140 Columns: 29
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (21): Category, MODIS_Tile, MYD13Q1_006__250m_16_days_VI_Quality_bitmas...
## dbl   (7): ID, Latitude, Longitude, MYD13Q1_006_Line_Y_250m, MYD13Q1_006_Sam...
## date  (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# filter for bad data
dfMYD = dfMYD[dfMYD$MYD13Q1_006__250m_16_days_VI_Quality_MODLAND_Description == "VI produced with good c
```

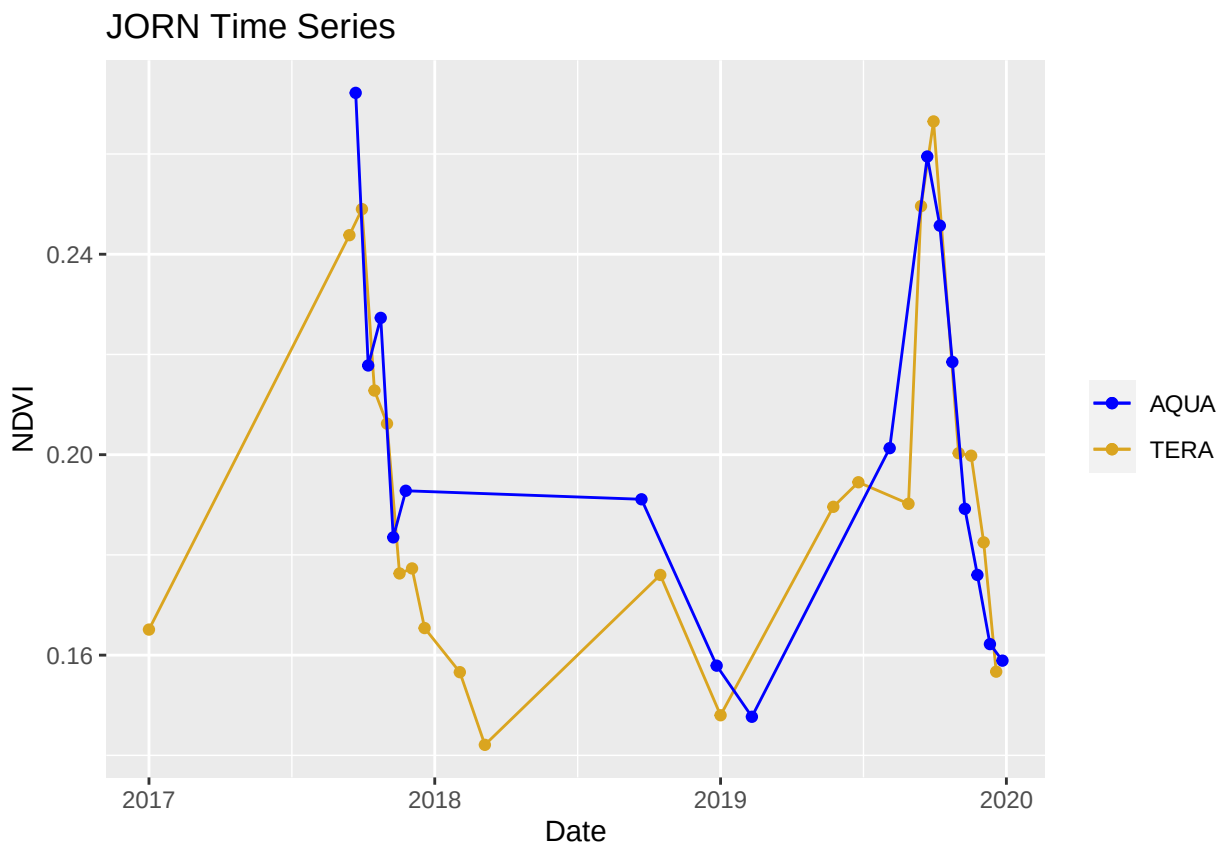
Question 4

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

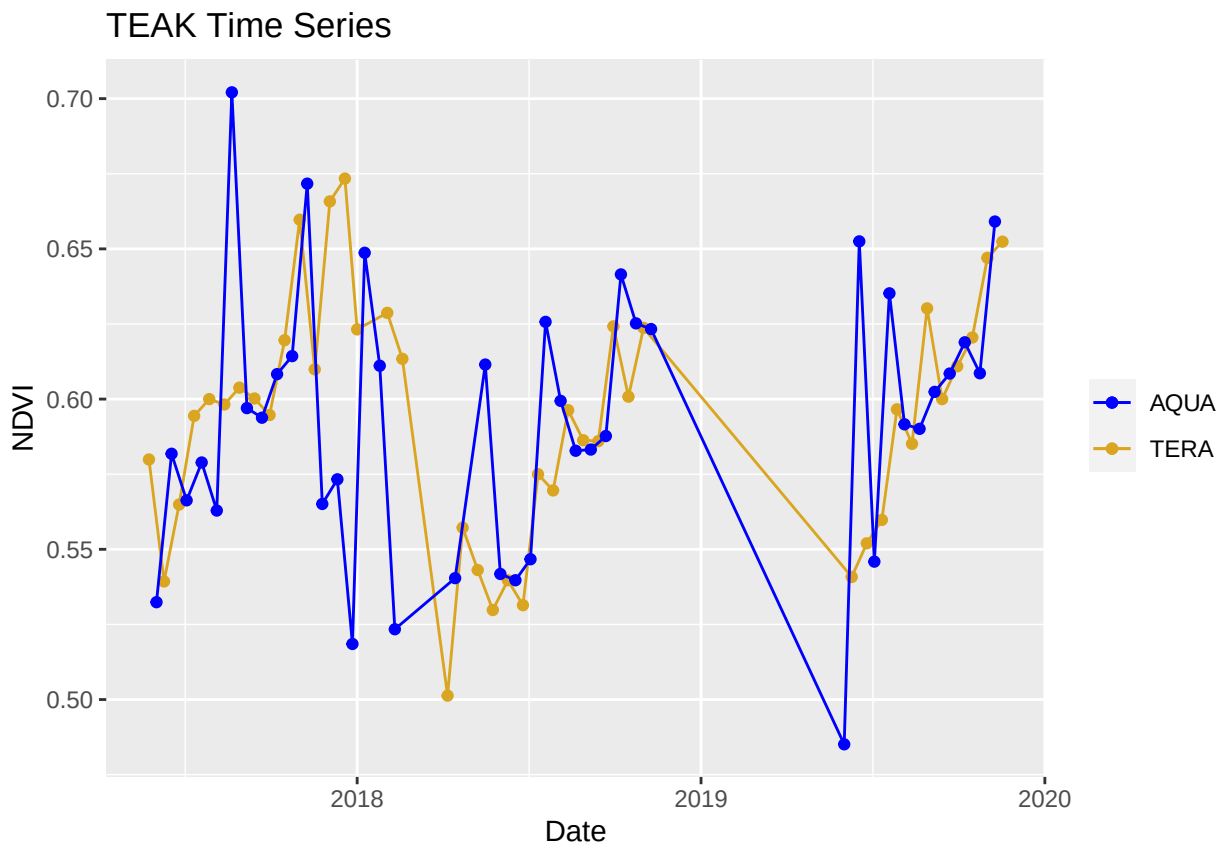
```
# split df's by site
dfmod_JORN <- dfMOD %>% filter( Category =='JORN')
dfmod_TEAK <- dfMOD %>% filter( Category =='TEAK')
dfmyd_JORN <- dfMYD %>% filter( Category =='JORN')
dfmyd_TEAK <- dfMYD %>% filter( Category =='TEAK')
```

```
# JORN
ggplot()+
  geom_line(data = dfmod_JORN,
            aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_JORN,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
```

```
  geom_point(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  scale_color_manual(name = NULL, values=c("blue", "goldenrod")) +
  labs(title = "JORN Time Series", x = "Date", y = "NDVI")
```




```
# TEAK
ggplot()+
  geom_line(data = dfmod_TEAK,
            aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_TEAK,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  scale_color_manual(name = NULL, values=c("blue", "goldenrod")) +
  labs(title = "TEAK Time Series", x = "Date", y = "NDVI")
```



Question 5

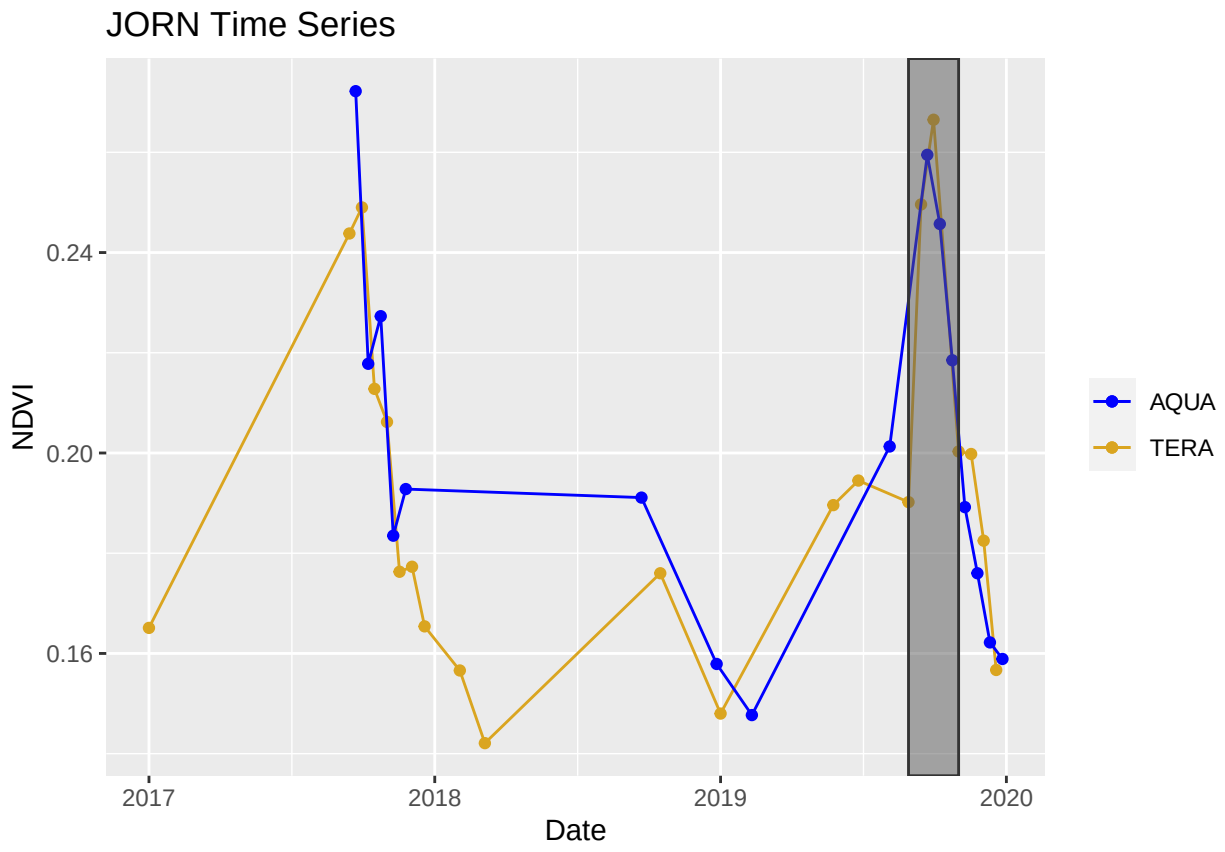
Constrain a 3-week window for 'peak greeness' from MODIS and highlight it on your timeseries plot.

```
# what is the date with the highest greenness
which.max(dfmod_JORN$MOD13Q1_006__250m_16_days_NDVI) # 17
```

```
## [1] 17
```

```
# make new object that is ~3 weeks JORN's peak greenness
rect = rect <- data.frame(xmin=dfmod_JORN$Date[15], xmax=dfmod_JORN$Date[18], ymin=-Inf, ymax=Inf)

# add this^ to timeseries plot
ggplot()+
  geom_line(data = dfmod_JORN,
            aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_JORN,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  scale_color_manual(name = NULL, values=c("blue", "goldenrod")) +
  labs(title = "JORN 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)
```

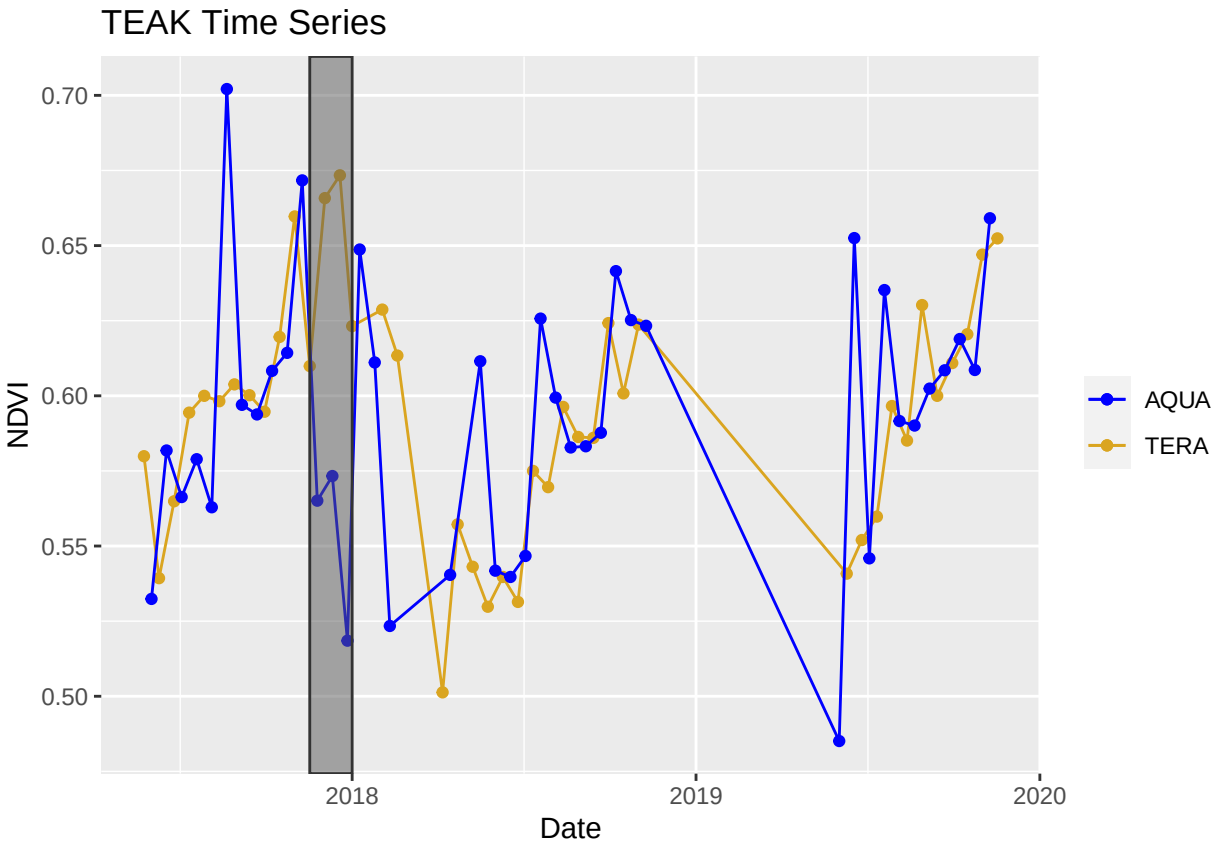



```
# im only going to consider TERA greennes for simplicity since aqua & tera are giving me different gree

# what is the date with the highest greenness
which.max(dfmod_TEAK$MOD13Q1_006__250m_16_days_NDVI) # 14
```

```
## [1] 14
```

```
# make new object that is ~3 weeks TEAK's peak greenness
rect = rect <- data.frame(xmin=dfmod_TEAK$Date[12], xmax=dfmod_TEAK$Date[15], ymin=-Inf, ymax=Inf)

# add this^ to timeseries plot
ggplot()+
  geom_line(data = dfmod_TEAK,
            aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_TEAK,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  scale_color_manual(name = NULL, values=c("blue", "goldenrod")) +
```

```
  labs(title = "TEAK 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)
```



Question 6

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

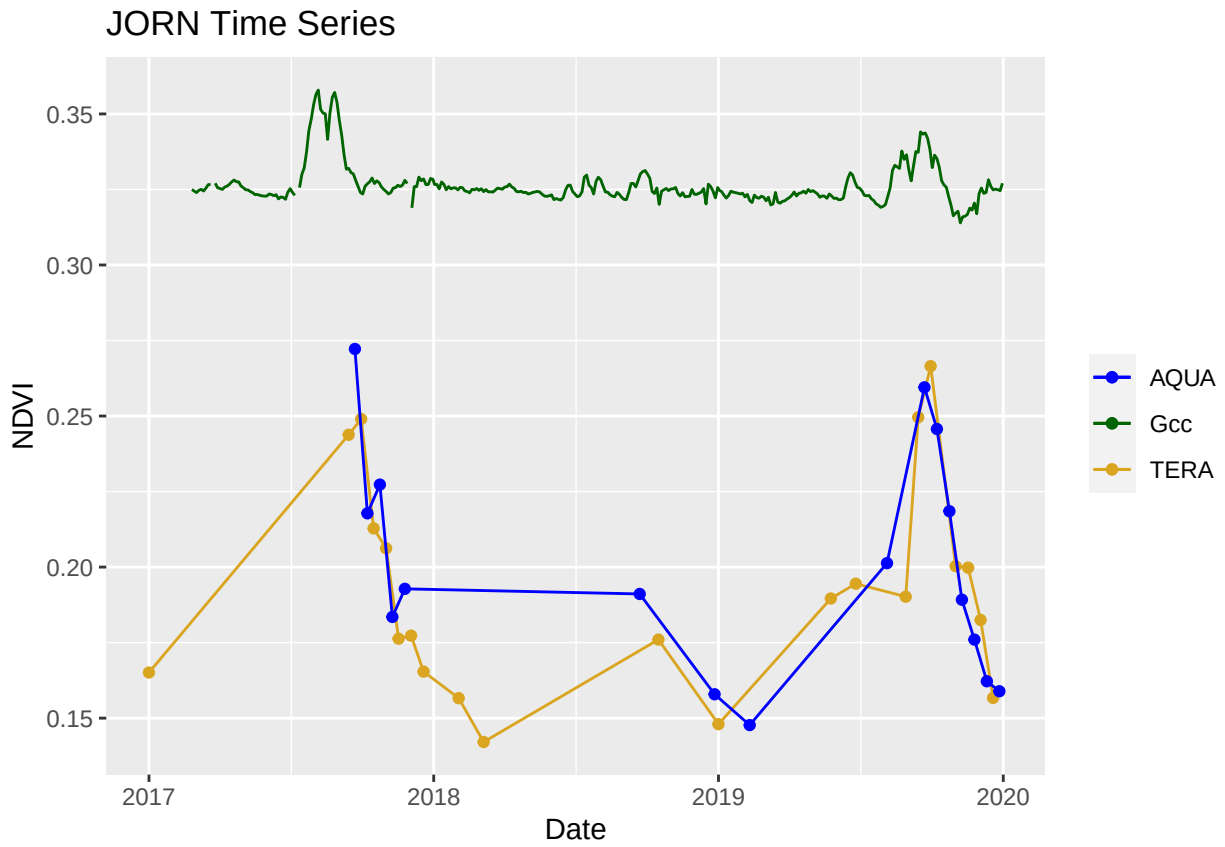
```
library(phenocamapi, quietly = T)
library(xROI, quietly = T)

# download phenocam data for JORN
JORN_1000 <- get_pheno_ts(site = 'NEON.D14.JORN.DP1.00033', vegType = 'GR', roiID = 1000, type = '3day'
# filter for the same date range
JORN_1000 = JORN_1000 %>%
  filter(date >= "2017-01-01",
         date <= "2019-12-31")

# download phenocam data for TEAK
TEAK_1000 <- get_pheno_ts(site = 'NEON.D17.TEAK.DP1.00033', vegType = 'EN', roiID = 1000, type = '3day'
# filter for the same date range
# filter for the same date range
```

```
TEAK_1000 = TEAK_1000 %>%
  filter(date >= "2017-01-01",
         date <= "2019-12-31")
```

Question 7

Plot the PhenoCam and MODIS timeseries on the same plot.

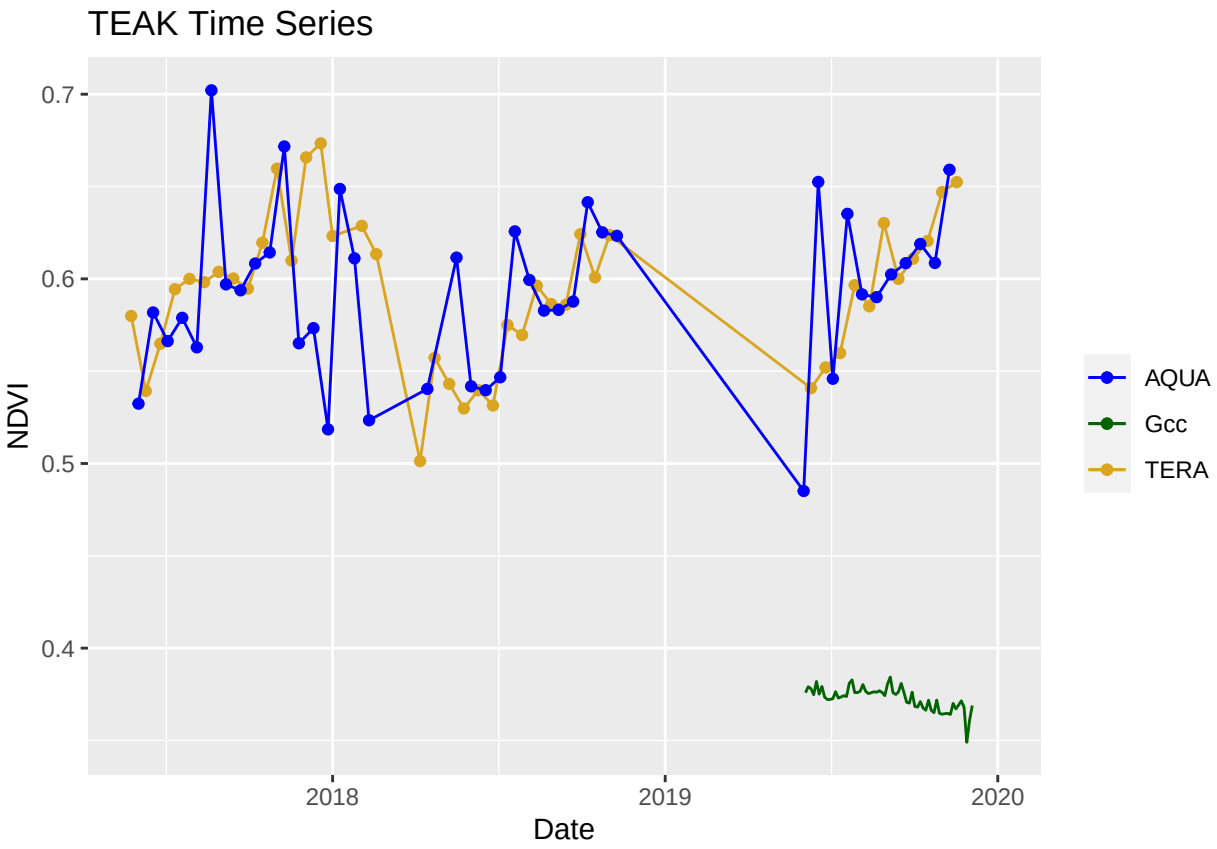
```
# JORN
ggplot()+
  geom_line(data = dfmod_JORN,aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_JORN,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  geom_line(data = JORN_1000, aes(x = as.Date(date), y = JORN_1000$gcc_90, color = "Gcc"))+
  scale_color_manual(name = NULL, values=c("blue", "darkgreen","goldenrod" )) +
  labs(title = "JORN Time Series", x = "Date", y = "NDVI")
```




```
# TEAK
ggplot()+
  geom_line(data = dfmod_TEAK,
            aes(x= Date,
```

```
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_TEAK,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  geom_line(data = TEAK_1000, aes(x = as.Date(date), y = gcc_90, color = "Gcc"))+
  scale_color_manual(name = NULL, values=c("blue", "darkgreen", "goldenrod")) +
  labs(title = "TEAK Time Series", x = "Date", y = "NDVI")
```

```
## Warning: Removed 9 row(s) containing missing values (geom_path).
```



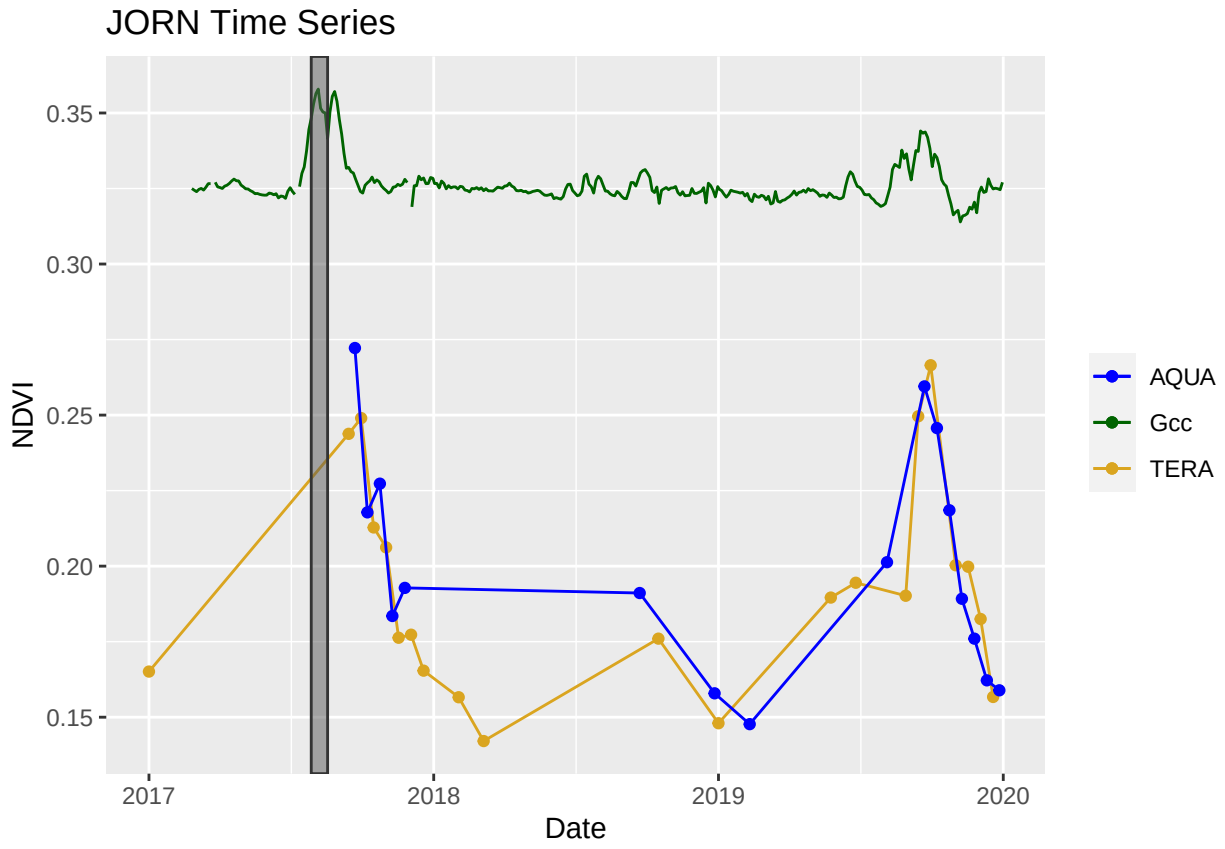
Question 8

Constrain a 3-week window for 'peak greeness' from PhenoCam and highlight it on your timeseries.

```
# which date/row is the highest greenness at JORN?
which.max(JORN_1000$gcc_90) ## 55
```

```
## [1] 55
```

```
# define a rectanlge around peak greenness
rectGccJORN = data.frame(xmin= as.Date(JORN_1000$date[52]), xmax=as.Date(JORN_1000$date[59]), ymin=-Inf
# add the plot
ggplot()+
  geom_line(data = dfmod_JORN,aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_JORN,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  geom_line(data = JORN_1000, aes(x = as.Date(date), y = JORN_1000$gcc_90, color = "Gcc"))+
  scale_color_manual(name = NULL, values=c("blue", "darkgreen","goldenrod" )) +
  labs(title = "JORN Time Series", x = "Date", y = "NDVI") +
  geom_rect(data=rectGccJORN, aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
            color="grey20", alpha=0.5,
            inherit.aes = FALSE)
```



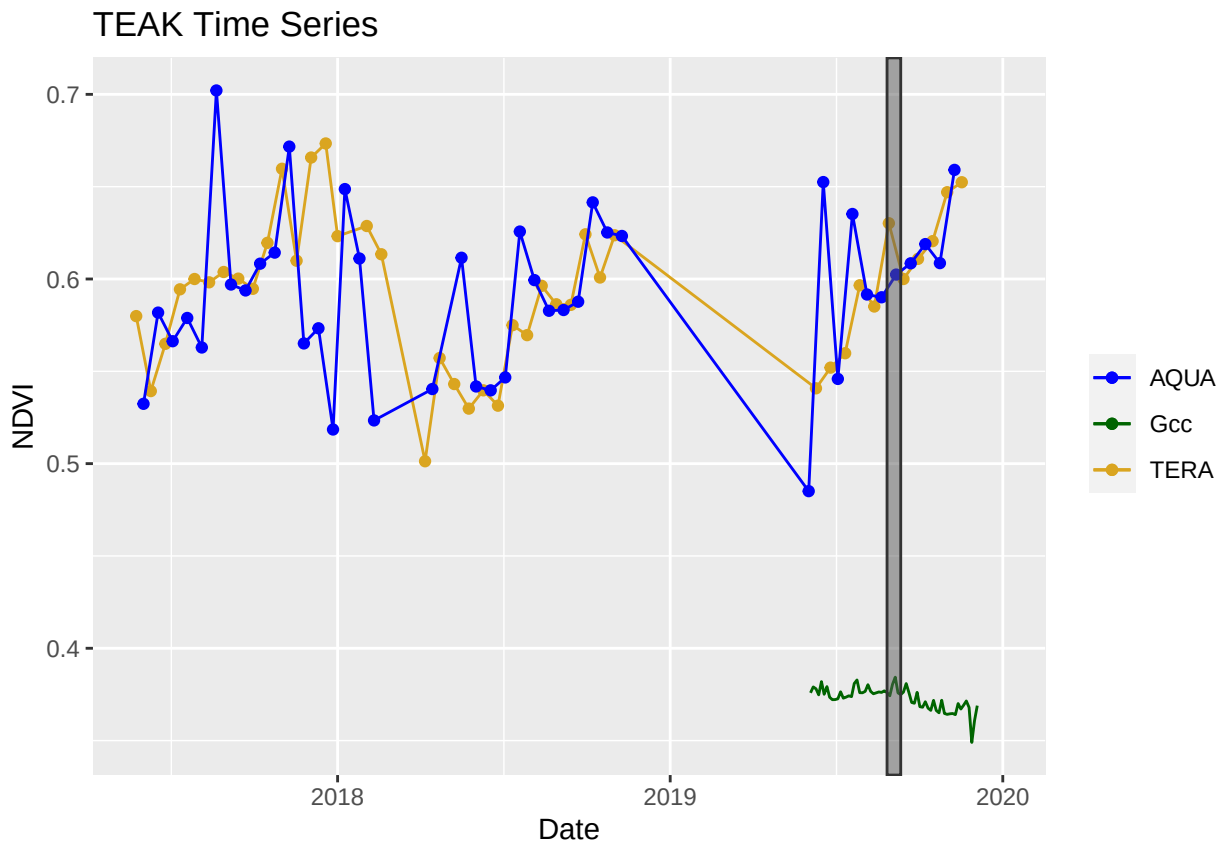

```
# which date/row is the highest greenness at TEAK?
which.max(TEAK_1000$gcc_90) ## 32
```

```
## [1] 32
```

```
# define a rectanlge around peak greenness
rectGccTEAK = data.frame(xmin= as.Date(TEAK_1000$date[29]), xmax=as.Date(TEAK_1000$date[34]), ymin=-Inf
```

```
# add the plot
ggplot()+
  geom_line(data = dfmod_TEAK,aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_TEAK,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  geom_line(data = TEAK_1000, aes(x = as.Date(date), y = TEAK_1000$gcc_90, color = "Gcc"))+
  scale_color_manual(name = NULL, values=c("blue", "darkgreen","goldenrod" )) +
  labs(title = "TEAK Time Series", x = "Date", y = "NDVI") +
  geom_rect(data=rectGccTEAK, aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
            color="grey20", alpha=0.5,
            inherit.aes = FALSE)
```

```
## Warning: Removed 9 row(s) containing missing values (geom_path).
```



Question 9

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

JORN flight dates (b/w 2017 & 2019):

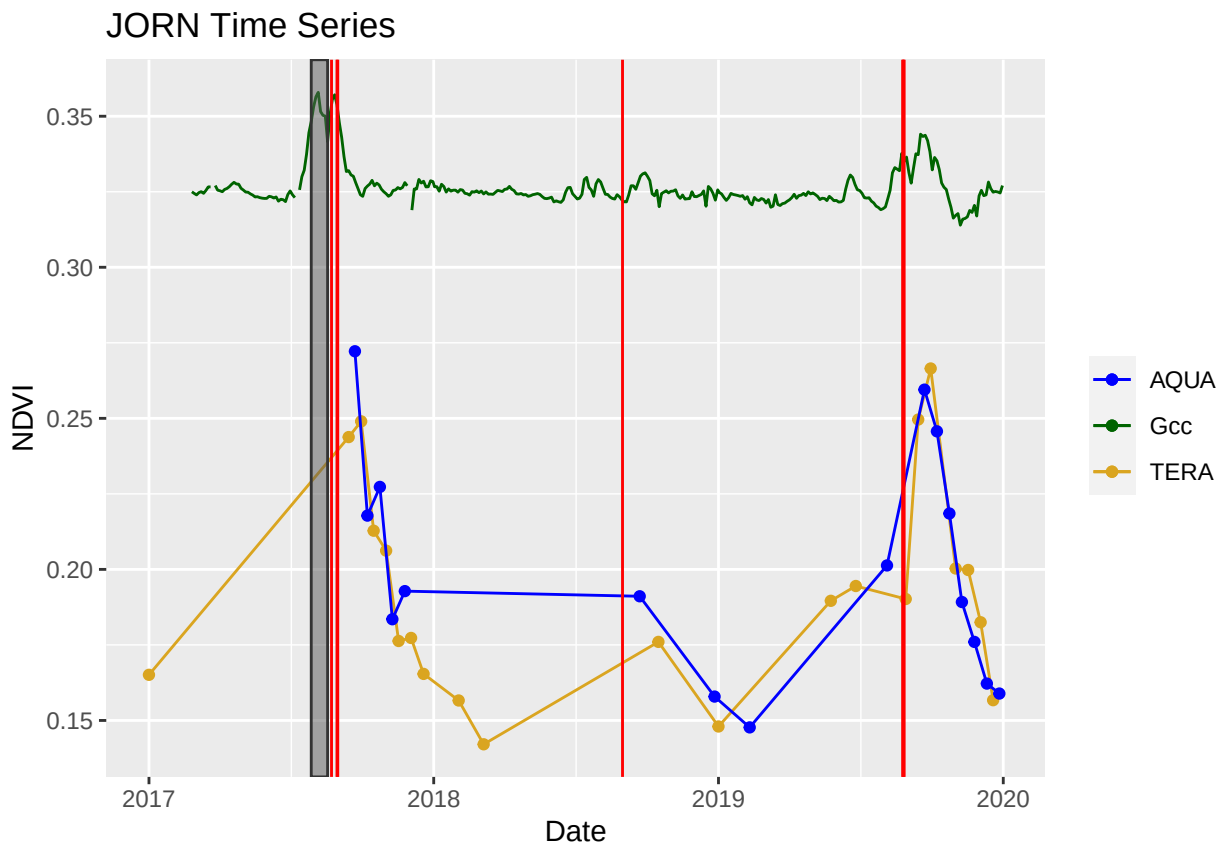
- 2017083115

- 2017082315
- 2017083015
- 2018083115
- 2019082615
- 2019082514
- 2019082715

TEAK flight dates:

- 2017062815
- 2017063016
- 2017062915
- 2017062715
- 2018061416
- 2018061515
- 2018061615
- 2019061715
- 2019061615
- 2019061515
- 2019061415

```
JORNflights = lubridate::ymd_h(c(2017083115,
2017082315,
2017083015,
2018083115,
2019082615,
2019082514,
2019082715))

ggplot()+
  geom_line(data = dfmod_JORN,aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_JORN,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_JORN,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  geom_line(data = JORN_1000, aes(x = as.Date(date), y = JORN_1000$gcc_90, color = "Gcc"))+
  scale_color_manual(name = NULL, values=c("blue", "darkgreen","goldenrod" )) +
  labs(title = "JORN Time Series", x = "Date", y = "NDVI") +
  geom_rect(data=rectGccJORN, aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
            color="grey20", alpha=0.5,
            inherit.aes = FALSE) +
  geom_vline(aes(xintercept = as.Date(JORNflights)), color = "red")
```




```
TEAKflights = lubridate::ymd_h(c(2017062815,
2017063016,
2017062915,
2017062715,
2018061416,
2018061515,
2018061615,
2019061715,
2019061615,
2019061515,
2019061415))

ggplot()+
  geom_line(data = dfmod_TEAK,aes(x= Date,
                y = MOD13Q1_006__250m_16_days_NDVI,
                color = "TERA"))+
  geom_point(data = dfmod_TEAK,aes(x= Date, y = MOD13Q1_006__250m_16_days_NDVI, color = "TERA")) +
  geom_line(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA")) +
  geom_point(data = dfmyd_TEAK,aes(x= Date, y = MYD13Q1_006__250m_16_days_NDVI, color = "AQUA"))+
  geom_line(data = TEAK_1000, aes(x = as.Date(date), y = TEAK_1000$gcc_90, color = "Gcc"))+
  scale_color_manual(name = NULL, values=c("blue", "darkgreen","goldenrod" )) +
  labs(title = "TEAK Time Series", x = "Date", y = "NDVI") +
  geom_rect(data=rectGccTEAK, aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
            color="grey20", alpha=0.5,
            inherit.aes = FALSE) +
  geom_vline(aes(xintercept = as.Date(TEAKflights)), color = "red")
```

```
## Warning: Removed 9 row(s) containing missing values (geom_path).
```