MODIS NDVI time series with BFAST

Jan Verbesselt

March 14, 2014

Abstract

This document explains how to use R scripting language for downloading MODIS data and analysing it within R. The results of the analysis of MODIS data within R are illustrated. For this time series analysis demonstration it is not required to know R details, we only use R for some practical demonstration of its great potential. In this exercise we will automatically download MODIS data for specific locations, i.e. Flux tower sites, around the world. First, an introduction to MODIS satellite data and the flux tower sites follow. Second, the use of R is introduced. Finally, the exercise in R is explained, step by step.

Contents

1 MODIS satellite data **2**

2 Online Analysis of MODIS satellite image data **2**

3 Getting started with R **3**

4 Install packages and define functions for MODIS data analysis **6**

5 Downloading MODIS data using R script **6**
5.1 Automatic MODIS data downloading 6
5.2 Manual MODIS data Downloading 7
5.3 The MODIS data structure 8

6 Visualising a modis time series above the fluxtower **8**
6.1 Plotting a MODIS NDVI time series 8
6.2 Plotting a MODIS Reliability time series and clean the NDVI data 11

7 Applying BFAST on cleaned NDVI time series **12**

8 Applying BFASTmonitor on the cleaned NDVI time series **15**

9 For a super Bonus (optional) **15**

10 More information **16**

1 MODIS satellite data

The MODIS satellite data that we will download for this time series analysis exercise is available from the following site: http://daac.ornl.gov/cgi-bin/MODIS/GR_col5_1/mod_viz.html. MODIS data is made available for subsets above a global network of flux towers. FLUXNET, a "network of regional networks", coordinates regional and global analysis of observations from micrometeorological tower sites. The flux tower sites use eddy covariance methods to measure the exchanges of carbon dioxide (CO_2), water vapor, and energy between terrestrial ecosystems and the atmosphere. The FLUXNET database contains information about tower location and site characteristics as well as data availability. More information above what a flux tower is and the network of flux towers can be found here: http://www.fluxnet.ornl.gov/fluxnet/index.cfm. For this exercise we will focus on the analysis of MODIS satellite data available for these flux towers. More specifically, we will look at the MODIS product called MOD13Q1 which are global 16-day images at a spatial resolution of 250 m. Each image contains several bands; i.e. blue, red, and near-infrared reflectances, centered at 469-nanometers, 645-nanometers, and 858-nanometers, respectively, are used to determine the MODIS vegetation indices. The MODIS Normalized Difference Vegetation Index (NDVI) complements NOAA's Advanced Very High Resolution Radiometer (AVHRR) NDVI products and provides continuity for time series historical applications. MODIS also includes a new Enhanced Vegetation Index (EVI) that minimises canopy background variations and maintains sensitivity over dense vegetation conditions. The EVI also uses the blue band to remove residual atmosphere contamination caused by smoke and sub-pixel thin cloud clouds. The MODIS NDVI and EVI products are computed from atmospherically corrected bi-directional surface reflectances that have been masked for water, clouds, heavy aerosols, and cloud shadows. Vegetation indices are used for global monitoring of vegetation conditions and are used in products displaying land cover and land cover changes. These data may be used as input for modeling global biogeochemical and hydrologic processes and global and regional climate. These data also may be used for characterizing land surface biophysical properties and processes, including primary production and land cover conversion. We will work with the MODIS NDVI band within the MOD13Q1 product. More information about this MODIS product can be found here: https://lpdaac.usgs.gov/products/modis_products_table/mod13q1. Go to the NDVI and the pixel reliability Layer information and have a look.

Question 1: By what factor does the 250m MODIS NDVI image layer need to be multiplied in order to obtain values between 0 and 1?

Question 2: What *rank key* (number?) would you use to obtain Good Data?

2 Online Analysis of MODIS satellite image data

Please go to the MODIS Land subsets website http://daac.ornl.gov/cgi-bin/MODIS/GR_col5_1/mod_viz.html:

1. Select Country: The Netherlands and select the Loobos Site.
2. Have a look at the *Corner coordinates and site details* (this will be important for the R script as explain below).
3. Via this site the data can be downloaded manually. We will automatically download the MODIS data via the R script.
4. Click on Time Series Advanced Version (User Defined QC setting) and select the MOD13Q1 data.
5. Look at the NDVI time series data and also click on the google maps link to investigate the land cover type. It is mainly forested by Pinus sylvestris or also called Scots Pine (within google maps satellite view).

Question 3: At which pixel number is the flux tower (i.e. the Site pixel) positioned in the MODIS 250m data grid (have a look at the link for corner coordinates and site details)

Question 4: What happens with the NDVI Filter Applied Graph if you select only data that you can use with Confidence?

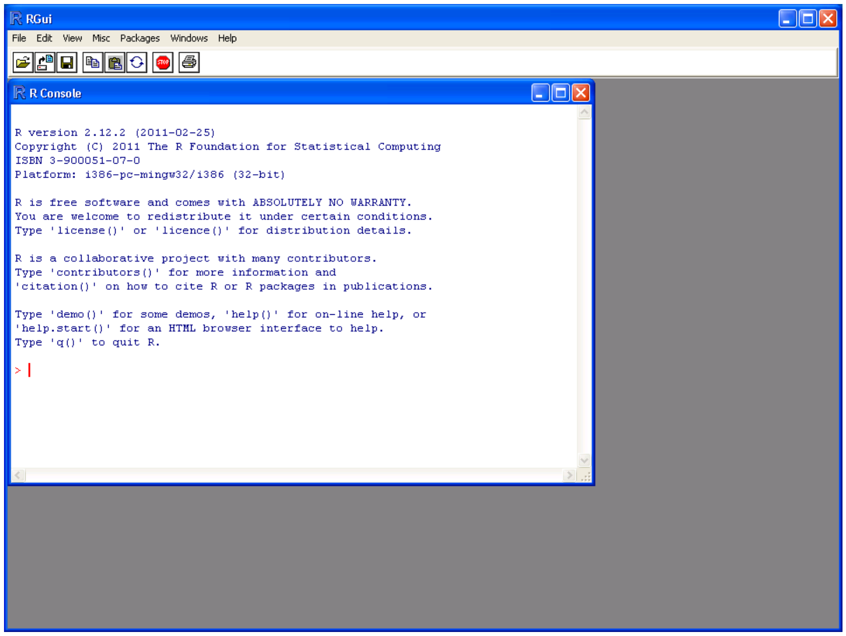

Figure 1: The graphical user interface to R

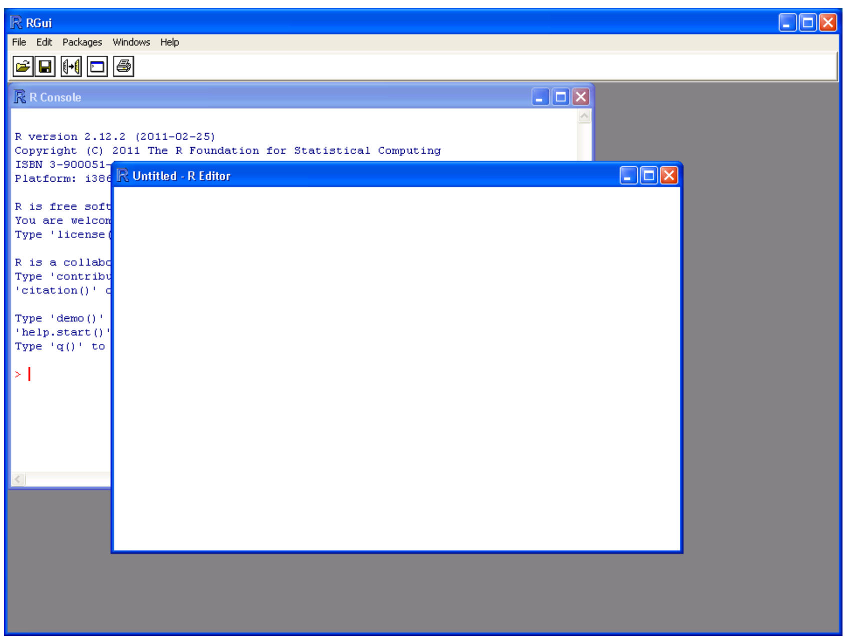

Figure 2: The graphical user interface with an empty script

3 Getting started with R

We will download the MODIS data for the Loobos Site via R and process the data for one location to detect changes within the time series. When you open R you will see Fig. 1. The window in the top left corner is the R console (e.g. statistical and spatial analysis tools). Go to the menu, click on File > new script and a script window will appear. The interface should now look something like Fig. 2.

You are now going to pass what you have written in your script to the console line by line and we will discuss what R is doing with your code. Select the first two lines with your mouse and then type **Ctrl-r**. The selected lines will be passed to the R console and your console should now look like something like this:

```
a <- 1
a
```

The first line you passed to the console created a new object named *a* in memory. The symbol '<-' is somewhat equivalent to an equal sign. In the second line you printed *a* to the console by simply typing it's name.

Now try to obtain he following output in the R console by writing the commands in the script window and running the via **Crtl-r**:

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

Now copy/paste the following script sections (in the grey zone) to your script window and run it step by step. The result is shown behind the # # sign:

```
class(a)

## [1] "numeric"
```

You now have requested the **class** attribute of *a* and the console has returned the attribute: **numeric**. R possesses a simple mechanism to support an object-oriented style of programming. All objects (*a* in this case) have a class attribute assigned to them. **R** is quite forgiving and will assign a class to an object even if you haven't specified one (as you didn't in this case). Classes are a very important feature of the **R** environment. Any function or method that is applied to an object takes into account its class and uses this information to determine the correct course of action. A simple example should suffice to explain further:

```
b <- 2
a + b

## [1] 3

newfunc <- function(x, y) {
    2 * x + y
}
a2b <- newfunc(2, 4)
a2b

## [1] 8
```

Select the next two lines using your mouse and pass these to the console using **Crtl-r**. The first line passed declares a new object *b*. The second line passed adds *a* and *b* together and prints the solution to the console. **R** has assessed the class attribute of *a* and *b*; determined they are both **numeric** objects, and; carried out the arithmetic calculation as requested.

The 4th line passed declares a new object **newfunc** (this is just a name and if you like you can give this function another name). It is a new function. Appearing in the first set of brackets is an argument list that specifies (in this case) two names. The value of the function appears within the second set of brackets where the process applied to the named objects from the argument list is defined.

Next, a new object *a2b* is created which contains the result of applying **newfunc** to the two objects you have defined earlier. The second last R command prints this new object to the console. Finally, you can now remove the objects you have created to make room for the next exercise by selecting and running:

```r
rm(a, b, newfunc, a2b)
```

R is supported by a very comprehensive help system. Help on any function can be accessed by entering the name of the function into the console preceded with a ?. The easiest way to access the system is to open a web-browser. This help system can be started by entering **help.start()** in the R console. Try it and see what happens.

```r
help(class)
```

For more information about R please refer to the following links http://www.statmethods.net/index.html. This is a great website for learning R function, graphs, and stats. Also visit http://www.r-project.org/ and check out the Manuals i.e an introductions to R. Welcome the Rrrrrr world!

4 Install packages and define functions for MODIS data analysis

Now we are ready to get started with the MODIS time series analysis exercise in R! First, the necessary add-on packages need to be installed within R in case there are not installed yet (so run the section below only once!):

```
## install these package in case they are not available on your computer
install.packages("strucchange")
install.packages("forecast")
install.packages("zoo")
install.packages("bfast", repos = "http://R-Forge.R-project.org", dependencies = TRUE)
```

In the next section below, **library(zoo)** loads a library with predefined time series analysis functions in R. This is the great thing about R, there are many other R packages available (for FREE!) that you can upload in R and make it more functional. For a overview of available packages is available here: http://crantastic.org/.

All the other lines of the section above need to be run at once (select all of them in the R script window and do **Crtl-r**), this will define two simple functions which we will need to process the MODIS data time series. So nothing will happen now in R, but the function are loaded and ready to be used in the script sections below.

```
## a function to create a regular 'ts' (time series) object in R using time information (dt)
timeser <- function(index, dt) {
    z <- zoo(index, dt)
    yr <- as.numeric(format(time(z), "%Y"))
    jul <- as.numeric(format(time(z), "%j"))
    delta <- min(unlist(tapply(jul, yr, diff)))  # 16
    zz <- aggregate(z, yr + (jul - 1)/delta/23)
    (tso <- as.ts(zz))
    return(tso)
}

## a function to remove values (set NA) that do not equal a certain criteria
sel <- function(y, crit) {
    ts.sel <- y
    ts.sel[!crit] <- NA
    return(ts.sel)
}
```

5 Downloading MODIS data using R script

Now we are ready to start downloading the MODIS data. There are two methods; (1) automatic downloading via the ftp server in the U.S. within R using the code section below (Section 5.1), (2) manual downloading of the data (Section 5.2). We will use the automatic downloading method for the exercise (easier;-)).

5.1 Automatic MODIS data downloading

We will use this method in the exercise as long as the server in the U.S. is online and working (you need internet connection, and also keep in mind that the file can be more than 15Mb large).

```
getwd()  ## the file is downloaded to your working directory
```

```
## you can set the working directory yourself via Rstudio if you go to 'session' and then
## 'set working directory'
fluxtoren <- c("fn_nlloobos.txt")
filenaam <- paste("ftp://daac.ornl.gov//data/modis_ascii_subsets//C5_MOD13Q1/data/MOD13Q1.",
    fluxtoren, sep = "")

## if the file exists already in your working directory nothing will happen:
if (!file.exists(fluxtoren)) {
    download.file(filenaam, fluxtoren)
    modis <- read.csv(fluxtoren, colClasses = "character")
} else {
    modis <- read.csv(fluxtoren, colClasses = "character")
}
## if the above step does not work you can download the data manually (go to 'Manual
## Downloading').
```

By running the lines above the MODIS data subset for the Loobos fluxtower (the Netherlands) is downloaded to the **modis** variable. Please be patient when running the code section above. Data for a different fluxtower, a fluxtower in New South Wales, Australia, can be downloaded by changing the **fluxtoren** variable using the following line:

```
fluxtoren <- c("fn_autumbar.txt")
```

Please try and change the name of the flux tower site and then rerun the section above to download data for another flux tower. The names of the flux towers for which MODIS data is available can be found in the following file available via this link; MODISSubsetSiteInformation, which you can open in Excel. The names that you need are in the $Site_{ID}$ column.

5.2 Manual MODIS data Downloading

If the above section does not work, you can download the data manually into your working directory via the following site: ModisViZ and can be loaded in R via the R script below. Things to do to download the data manually:

- download the .txt file from the MODIS Land Subsets website mentioned above, go to the e.g. Loobos site, click download the ASCII file, do *save as txt* file to save to a local folder, and rename the file to e.g. **NDVIMOD13Q1**.
- Read the data from R with the following R script lines.

First set your workdirectory using the **setwd()** command. Remark: on your computer the file path looks different on windows! In R you have to change the backslash symbol to a forward slash symbol.

```
## 'c:\student\MODIS'
setwd(c("c:/student/MODIS/"))
getwd()  ## to check what your working directory is.
```

Once your working directory is set correctly, you can read in the data file using the following command. Now the MODIS data is loaded!

```
modis <- read.csv("NDVIMOD13Q1.txt")
```

5.3 The MODIS data structure

The MODIS data within the 'modis' variable is organised so that the first six columns of the file contain information about filename, product (i.e. MOD13Q1), date (date of the image), Site (e.g. Loobos), Process-data, and band (i.e. one MODIS image has different bands = LAYERS, e.g. NDVI). For More information about the MODIS data look at: MODIS product table.

```
str(modis[1, 1:7])

## 'data.frame':	1 obs. of  7 variables:
##  $ HDFname    : chr "MOD13Q1.A2000049.fn_nlloobos.005.2006269104153.250m_16_days_blue_reflectance"
##  $ Product    : chr "MOD13Q1"
##  $ Date       : chr "A2000049"
##  $ Site       : chr "fn_nlloobos"
##  $ ProcessDate: chr "2006269104153"
##  $ Band       : chr "250m_16_days_blue_reflectance"
##  $ X1         : chr "148"
```

The following R names() shows the names of the first 8 columns of the 'modis' variable containing all the data.

```
names(modis)[1:8]

## [1] "HDFname"     "Product"     "Date"        "Site"        "ProcessDate" "Band"
## [7] "X1"          "X2"
```

The first six columns contain info about the image (e.g. site, date, band, and when it is processed) and from the 7th each column contains information about each MODIS pixel within the subset. E.g the 7th column is a pixel, and the 8th is another pixel. This shows the band names of this file. The MOD13Q1 Product contains 12 Bands:

```
modis$Band[1:12]

##  [1] "250m_16_days_blue_reflectance"          "250m_16_days_composite_day_of_the_year"
##  [3] "250m_16_days_EVI"                       "250m_16_days_MIR_reflectance"
##  [5] "250m_16_days_NDVI"                      "250m_16_days_NIR_reflectance"
##  [7] "250m_16_days_pixel_reliability"         "250m_16_days_red_reflectance"
##  [9] "250m_16_days_relative_azimuth_angle"    "250m_16_days_sun_zenith_angle"
## [11] "250m_16_days_view_zenith_angle"         "250m_16_days_VI_Quality"
```

6 Visualising a modis time series above the fluxtower

6.1 Plotting a MODIS NDVI time series

We select band 5 i.e. the NDVI band, and band 7 i.e. the band with reliability information

```
ndvibandname <- modis$Band[5]
rel <- modis$Band[7]
```

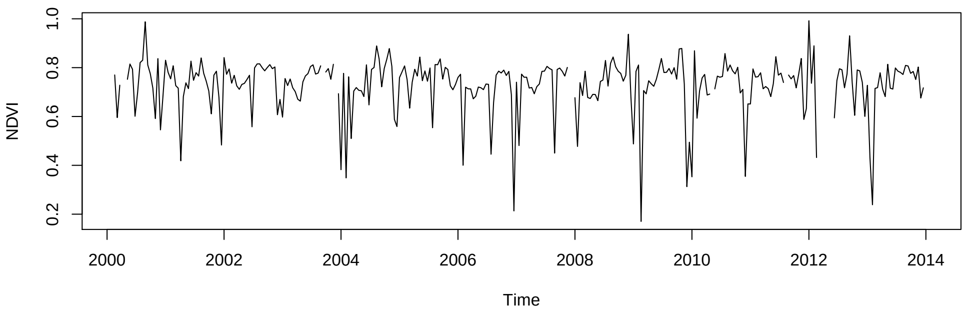

Figure 3: NDVI time series at the Flux towersite

We will select data for the pixel above the Loobos Fluxtower. Have a look at LoobosSiteInfo.

It is pixel number 436. Each column after the 6th column in the MODIS file contain data of one pixel so to select the data above the flux tower we have to add 6 to select the correct column within the matrix. The code section below will select the MODIS data for one pixel, scale the NDVI data by dividing it by 10000, and then plot the resulting variable **ts.NDVI** using the **plot()** function:

```
j <- (436) + 6  # we are adding 6 since the first data column is the 7th column
reliability <- as.numeric(modis[modis$Band == rel, j])  # reliability data
NDVI <- as.numeric(modis[modis$Band == ndvibandname, j])  # NDVI data
DATUM <- modis[modis$Band == ndvibandname, 3]  # dates
DATUM <- as.Date(DATUM, "A%Y%j")  # convert to a datum type
```

Now, let's create a time series! The NDVI value need to be scaled between 0-1 by dividing them by 10000. Attention! The 'Zoo' package is needed within the 'timeser' function so we load the packag using the line below.

```
library(zoo)  ## load the package

##
## Attaching package:  'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

ts.rel <- timeser(reliability, DATUM)
ts.NDVI <- timeser(NDVI/10000, DATUM)
```

Now plot the resulting **ts.NDVI** object (See Figure 3).

```
plot(ts.NDVI, ylab = "NDVI")
```

Question 5: What would happen if you would select multiple pixels (e.g. 3 pixels) and derive an average,maximum, and median of the 3 time series? Now make a plot showing the average, maximum, or median of the 3 NDVI time series. Compare the median NDVI time series with the NDVI time series of the flux tower and copy paste the R plot output in a word document.

What is the best technique to reduce the noise within a time series? How many pixels would you recommend selecting to reduce the noise level?

For bonus points and an extra challenge. Try to derive a 'noise reduced' NDVI time series of a 3 by 3 window around the flux tower.

```
## this is an example for two pixels try it out and customize for your own needs
j <- 442:444
t <- modis[modis$Band == ndvibandname, j]  # extract NDVI data
tt <- data.matrix(t)/10000  ## convert to a data matrix and divide by 10000
ttt <- ts(apply(tt, 2, timeser, DATUM), start = c(2000, 4), freq = 23)
## convert to a regular time series object plot(ttt) ## plot all the time series derive the
## statistics (max, mean):
maxt <- ts(apply(ttt, 1, max, na.rm = TRUE), start = c(2000, 4), freq = 23)
meant <- ts(apply(ttt, 1, mean, na.rm = TRUE), start = c(2000, 4), freq = 23)
## plot
plot(maxt, col = "green", ylim = c(0, 1))
lines(meant, col = "red")
##
```

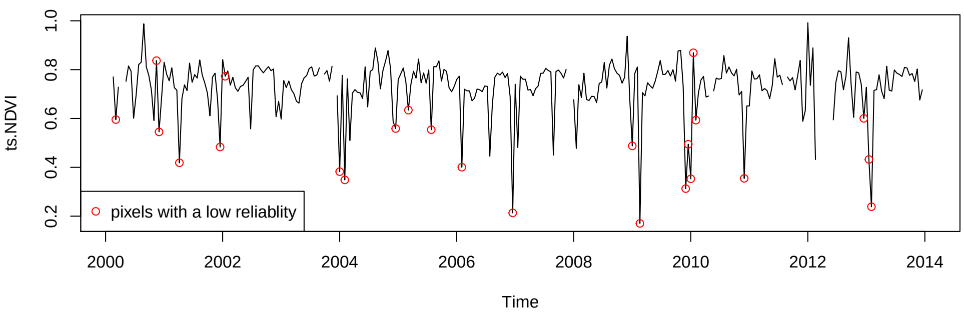

Figure 4: MODIS NDVI time series showing pixels with a low reliability.

6.2 Plotting a MODIS Reliability time series and clean the NDVI data

Now, we will visualize MODIS reliability information (See Figure 4). This R function plots a red point on the plot for all the data points in the time series with a reliability > 1. You can choose ts.rel = 1, or ts.rel > 2, or ..., and rerun the plot command again and see what happens.

```
plot(ts.NDVI)
lines(sel(ts.NDVI, ts.rel > 1), col = "red", type = "p")
legend("bottomleft", "pixels with a low reliablity", col = 2, pch = 1)
```

Question 6: Investigation of MODIS reliability scores. What happens if you select only good quality NDVI data? Can you explain what happens and why this could be? Discuss

Perform the cleaning and plot the result by running the following lines. The resulting plot will show the MODIS NDVI time series showing red section which indicate the zones that are deleted based on reliability information.

```
ts.clNDVI <- ts.NDVI
ts.clNDVI[ts.rel > 1] <- NA  # delete data with reliability > 1
```

By applying the two R script lines above, we set all the points with a reliability above 1 to NA (i.e. Not Available which is similar as deleting the value) in the **ts.clNDVI** variable. Now, plot the result of the cleaning and compare with the non-cleaned time series (See Figure 5):

```
plot(ts.NDVI, col = "red")
lines(ts.clNDVI, lwd = 2, col = "black")
```

There are still clouds effects visible in the NDVI time series after using the MODIS reliability information. The reliability information available with each MODIS image indicates how reliable the data is and is based on the cloud masking results, atmospheric data (aerosol thickness), satellite viewing angle, etc. More information about the reliability is available via the MODIS Product Table website.

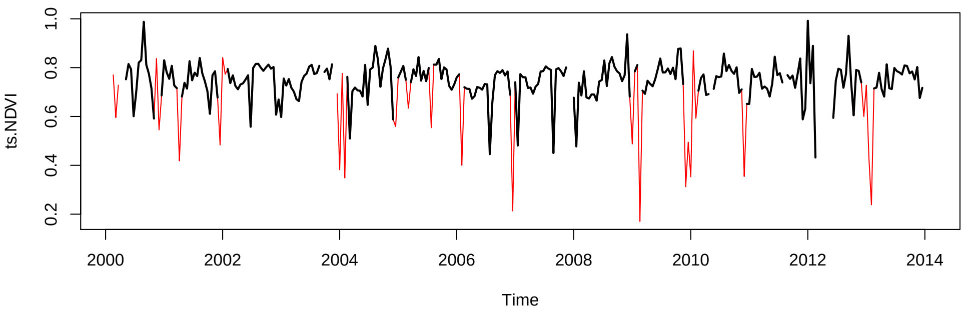

Figure 5: MODIS NDVI time series still showing remaining cloud effects.

7 Applying BFAST on cleaned NDVI time series

In this section we will use BFAST on the cleaned NDVI time series to detect changes within the time series. If the following R packages are not installed yet, they can be installed using the following commands.

```
## install these package in case they are not available on your computer
install.packages("strucchange")
install.packages("forecast")
install.packages("bfast", repos = "http://R-Forge.R-project.org")
```

The BFAST package can be loaded via:

```
library("bfast")

## Loading required package:  strucchange
## Loading required package:  sandwich
## Loading required package:  raster
## Loading required package:  sp
```

First, we will interpolate the gaps in the cleaned NDVI time series using a simple linear interpolation approach. The function that we use for this is the **na.aprox()** function which looks for NA's (Not Available's), which means dates for which no data is available (e.g., that we removed in the previous steps) and interpolates the data. The **plot()** command of the results (**ts.clNDVIfilled**) visualizes the result of the interpolation (See Figure 6).

```
ts.clNDVIfilled <- na.approx(ts.clNDVI)
plot(ts.clNDVIfilled, ylim = c(0.1, 1))
```

Second, we apply the BFAST function onto the time series. We determine this minimum distance between potentially detected breaks. Here, we set the distance to 25 time steps (i.e. 25 16-day images). Then we apply the BFAST function (bfast()) on the time series (Figure 7).

```
rdist <- 25/length(ts.clNDVIfilled)
## ratio of distance between breaks (time steps) and length of the time series
fit <- bfast(ts.clNDVIfilled, h = rdist, season = "harmonic", max.iter = 1)
plot(fit, main = "")
```

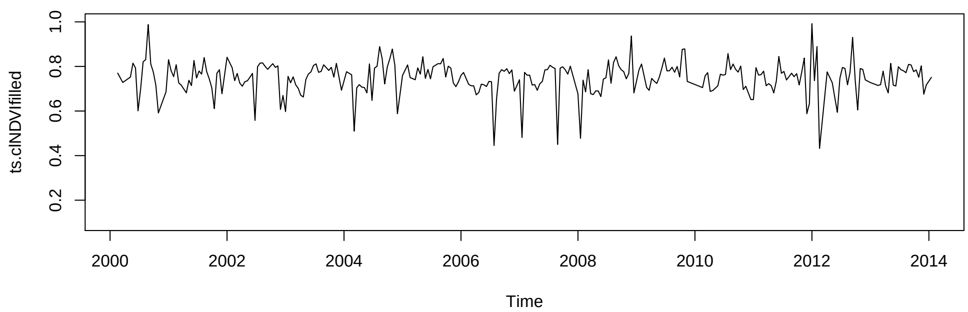

Figure 6: An NDVI time series without gaps.

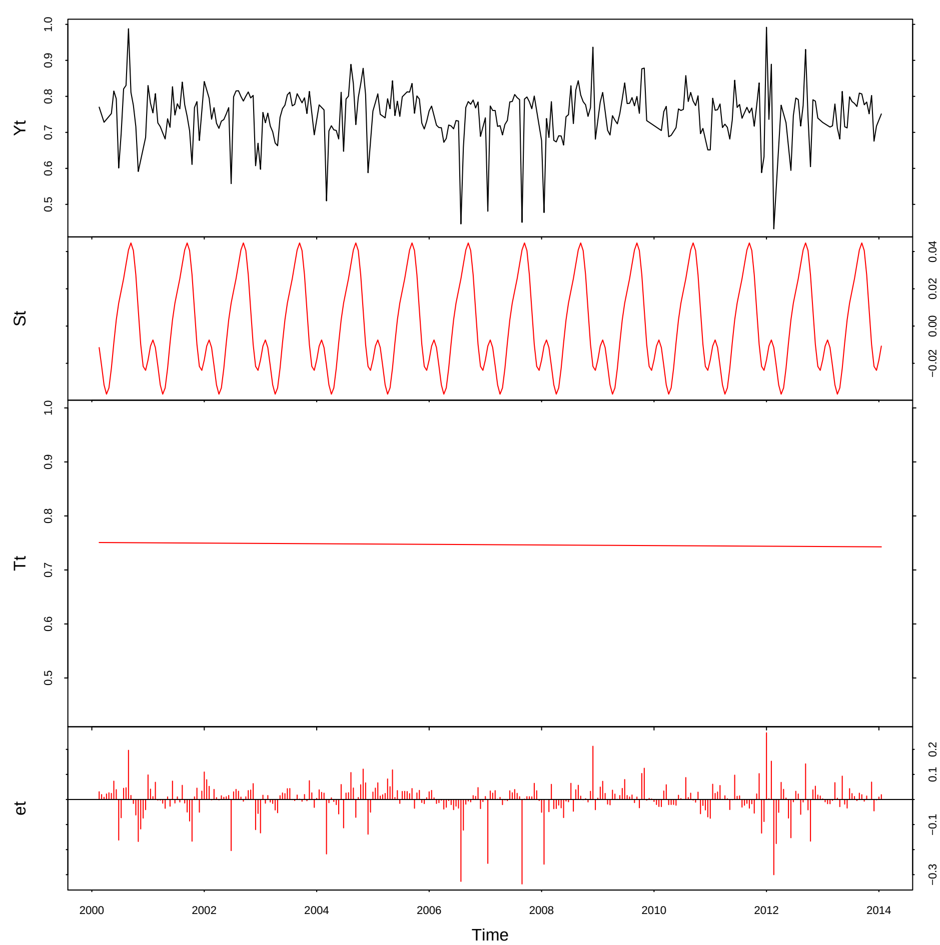

Figure 7: BFAST analysis of the cleaned and interpolated NDVI time series

Question 7: Copy past the resulting R BFAST graph in the report and describe the detected components and change types, detected within the image. Do you detect breaks? How strict would you do the cleaning?

Question 8: Download data from another location on earth and run all the steps mentioned above again in order to apply the BFAST function again onto a new cleaned NDVI time series. Plot the BFAST result graph in a word document, mention the flux tower that you downloaded the data from and describe the difference with the graph obtained from Question 7.

To better understand how bfastmonitor works have a look at the help section of bfastmonitor function and try out the examples provided.

```
help(bfast)
## for more info try out the examples in the bfast help section!
plot(harvest, ylab = "NDVI")  # MODIS 16-day cleaned and interpolated NDVI time series
(rdist <- 10/length(harvest))
# ratio of distance between breaks (time steps) and length of the time series
fit <- bfast(harvest, h = rdist, season = "harmonic", max.iter = 1, breaks = 2)
plot(fit)
## plot anova and slope of the trend identified trend segments
plot(fit, main = "")
```

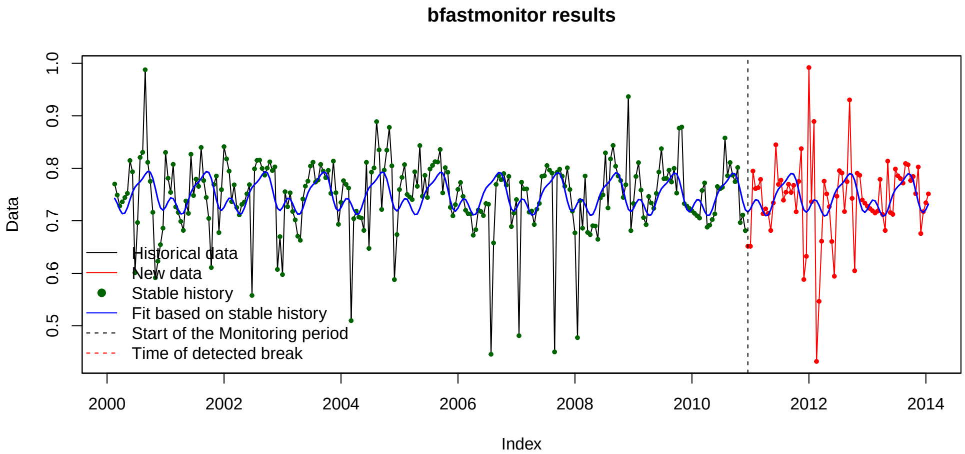

Figure 8: BFASTmonitor analysis of the cleaned and interpolated NDVI time series

8 Applying BFASTmonitor on the cleaned NDVI time series

To better understand how bfastmonitor works have a look at the help section of bfastmonitor function and try out the examples provided (Figure 8). For extra background information (Verbesselt et al., 2012).

```
mon <- bfastmonitor(ts.clNDVIfilled, start = c(2010, 23), formula = response ~ harmon + trend,
    history = c("ROC"))
plot(mon, main = "bfastmonitor results")
```

Question 9: How many stable years do you have in your history period? Illustrate this with your own time series from a flux tower of your own choice.

Question 10: Start the monitoring period the end of 2011. Is 2012 an abnormal year? Illustrate this with your own time series from a flux tower of your own choice.

Question 11: See the help section of bfastmonitor. Can you explain what the effect is of using a different "formula" in bfastmonitor(). For example, what happens if you use $response \sim trend$**. Illustrate this with your own time series from a flux tower of your own choice.**

```
`?`(bfastmonitor)
```

9 For a super Bonus (optional)

(For those who already followed the **Geo-scripting course**) Can somebody georeference the subset of data downloaded for a fluxtower and e.g. make a spatial map? Hint: The name and georeference information of the flux towers for which MODIS data is available can be found in the following file available via this link; MODISSubsetSiteInformation. Select all the NDVI data at one time step (e.g. the first measurement of 2011) and create raster from that.

```
library(raster)
```

```
`?`(projection)

## the site coordinates are projected in lat/long
latlong <- "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
## convert these to sinusoidal projection
modissinprojection <- "+proj=sinu +lon_0=0 +x_0=0 +y_0=0 +a=6371007.181 +b=6371007.181 +units=m +no_defs
## and then define your modis spatial point data frame
resolution <- 250
## define the extent by looking at the subset site information (xmin, xmax, ymin,ymax)
```

10 More information

More information can be found on the following website `http://bfast.r-forge.r-project.org/` and in the BFAST papers mentioned on the website.

References

Verbesselt, J., Zeileis, A., & Herold, M. (2012). Near real-time disturbance detection using satellite image time series. *Remote Sensing of Environment*, *123*, 98–108.