dtwSat: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R

Victor Maus
University of Münster
INPE, IIASA

Gilberto Câmara
INPE
University of Münster

Marius Appel
University of Münster

Edzer Pebesma
University of Münster

Abstract

The opening of large archives of satellite data such as LANDSAT, MODIS and the SENTINELs has given researchers unprecedented access to data, allowing them to better quantify and understand local and global land change. The need to analyse such large data sets has lead to the development of automated and semi-automated methods for satellite image time series analysis. However, few of the proposed methods for remote sensing time series analysis are available as open source software. In this paper we present the R package **dtwSat**. This package provides an implementation of the Time-Weighted Dynamic Time Warping method for land cover mapping using sequence of multi-band satellite images. Methods based on dynamic time warping are flexible to handle irregular sampling and out-of-phase time series, and they have achieved significant results in time series analysis. **dtwSat** is available from the Comprehensive R Archive Network and contributes to making methods for satellite time series analysis available to a larger audience. The package supports the full cycle of land cover classification using image time series, ranging from selecting temporal patterns to visualising and assessing the results.

Keywords: dynamic programming, MODIS time series, land cover changes, crop monitoring.

1. Introduction

Remote sensing images are the most widely used data source for measuring land use and land cover change (LUCC). In many areas, remote sensing images are the only data available for this purpose (Lambin and Linderman 2006; Fritz *et al.* 2013). Recently, the opening of large archives of satellite data such as LANDSAT, MODIS and the SENTINELs has given researchers unprecedented access to data, allowing them to better quantify and understand local and global land change. The need to analyse such large data sets has lead to the development of automated and semi-automated methods for satellite image time series analysis. These methods include multi-image compositing (Griffiths *et al.* 2013), detecting forest disturbance and recovery (Kennedy *et al.* 2010; Zhu *et al.* 2012; DeVries *et al.* 2015), crop classification (Xiao *et al.* 2005; Wardlow *et al.* 2007; Petitjean *et al.* 2012; Maus *et al.* 2016), planted forest mapping (le Maire *et al.* 2014), crop expansion and intensification (Galford *et al.* 2008; Sakamoto *et al.* 2009), detecting trend and seasonal changes (Lunetta *et al.* 2006; Verbesselt *et al.* 2010a,b, 2012), and extracting seasonality metrics from satellite time series (Jönsson and Eklundh 2002, 2004). Given the open availability of large image data sets, the Earth Observation community would get much benefit from methods that are openly available,

reproducible and comparable. However, few of the proposed methods for remote sensing time series analysis are available as open source software, the main exception being the BFAST and BFAST-monitor algorithms for change detection (Verbesselt *et al.* 2010a,b). This paper is a contribution to making methods for satellite time series analysis available to a larger audience.

In this paper we describe the **dtwSat** package, written in R (R Core Team 2016) and Fortran programming languages, and available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=dtwSat`. The package provides an implementation of Time-Weighted Dynamic Time Warping (TWDTW) (Maus *et al.* 2016) for satellite image time series analysis.

The TWDTW method is an adaptation of the well-known dynamic time warping (DTW) method for time series analysis (Velichko and Zagoruyko 1970; Sakoe and Chiba 1971, 1978; Rabiner and Juang 1993; Berndt and Clifford 1994; Keogh and Ratanamahatana 2005; Müller 2007) for land cover classification. The standard DTW compares a temporal signature of a known event (*e.g.*, a person's speech) with an unknown time series. It finds all possible alignments between two time series and provides a dissimilarity measure (Rabiner and Juang 1993). In contrast to standard DTW, the TWDTW method is sensitive to seasonal changes of natural and cultivated vegetation types. It also considers inter-annual climatic and seasonal variability. In a tropical forest area, the method has achieved a high accuracy for mapping classes of single cropping, double cropping, forest, and pasture (Maus *et al.* 2016).

We chose R because it is an open source software that offers a large number of reliable packages. The **dtwSat** package builds upon on a number of graphical and statistical tools in R: **dtw** (Giorgino 2009), **proxy** (Meyer and Buchta 2015), **zoo** (Zeileis and Grothendieck 2005), **mgcv** (Wood 2000, 2003, 2004, 2006, 2011), **sp** (Pebesma and Bivand 2005; Bivand *et al.* 2013), **raster** (Hijmans 2015), **caret** (Kuhn *et al.* 2016), and **ggplot2** (Wickham 2009). Other R packages that are related and useful for remote sensing and land cover analysis include **landsat** (Goslee 2011), **rgdal** (Bivand and Lewin-Koh 2015), **spacetime** (Pebesma 2012; Bivand *et al.* 2013), **bfast** (Verbesselt *et al.* 2010a,b), **bfastmonitor** (Verbesselt *et al.* 2011), **bfastSpatial** (Dutrieux and DeVries 2014), **MODISTools** (Tuck *et al.* 2014), **maptools** (Bivand and Lewin-Koh 2015), and **lucc** (Moulds *et al.* 2015). Using existing packages as building blocks, software developers in R save a lot of time and can concentrate on their intended goals.

There is already an R package that implements the standard DTW method for time series analysis: the **dtw** package (Giorgino 2009). In the **dtwSat** package, we focus on the specific case of satellite image time series analysis. The analysis method implemented in **dtwSat** package extends that of the **dtw** package; it adjusts the standard DTW method to account for the seasonality of different types of land cover. Our aim is to support the full cycle of land cover classification, from selecting sample patterns to visualising and assessing the final result.

This paper focuses on the motivation and guidance for using the TWDTW method for remote sensing applications. The full description of the method is available in a paper published by the lead author (Maus *et al.* 2016). In what follows, the section 2 describes the application of TWDTW (Maus *et al.* 2016) for satellite time series analysis. The section 3 gives an overview of the **dtwSat** package. Then, section 4 focuses on the analysis of a single time series and shows some visualisation methods. We then present an example of a complete land cover change analysis for a study area in Mato Grosso, Brazil in section 5.

2. The Time-Weighted Dynamic Time Warping method

In this section, we describe the Time-Weighted Dynamic Time Warping (TWDTW) algorithm in general terms. For a detailed technical explanation, refer to Maus *et al.* (2016). TWDTW is time-constrained version of the Dynamic Time Warping (DTW) algorithm. Although the standard DTW method is good for shape matching (Keogh and Ratanamahatana 2005), it is not suited *per se* for satellite image time series analysis, since it disregards the temporal range when finding the best matches between two time series (Maus *et al.* 2016). When using image time series for land cover classification, one needs to balance between shape matching and temporal alignment, since each land cover class has a distinct phenological cycle associated with the vegetation (Reed *et al.* 1994, Zhang *et al.* (2003)). For example, soybeans and maize cycles range from 90 to 120 days, whereas sugar-cane has a 360 to 720 days cycle. A time series with cycle larger than 180 days is unlikely to come from soybeans or maize. For this reason, Maus *et al.* (2016) include a time constraint in DTW to account for seasonality. The resulting method is capable of distinguishing different land cover classes.

The inputs to TWDTW are: (a) a set of time series of known temporal patterns (*e.g.*, phenological cycles of land cover classes); (b) an unclassified long-term satellite image time series. For each temporal pattern, the algorithm finds all matching subintervals in the long-term time series, providing a dissimilarity measure (cf. Figure 1). The result of the algorithm is a set of subintervals, each associated with a pattern and with a dissimilarity measure. We then break the unclassified time series in periods according to our needs (*e.g.*, yearly, seasonality, monthly). For each period, we consider all matching subintervals that intersect with it, and classify them based on the land cover class of the best matching subinterval. In this way, the long-term satellite time series is divided in periods, and each period is assigned a land cover class.

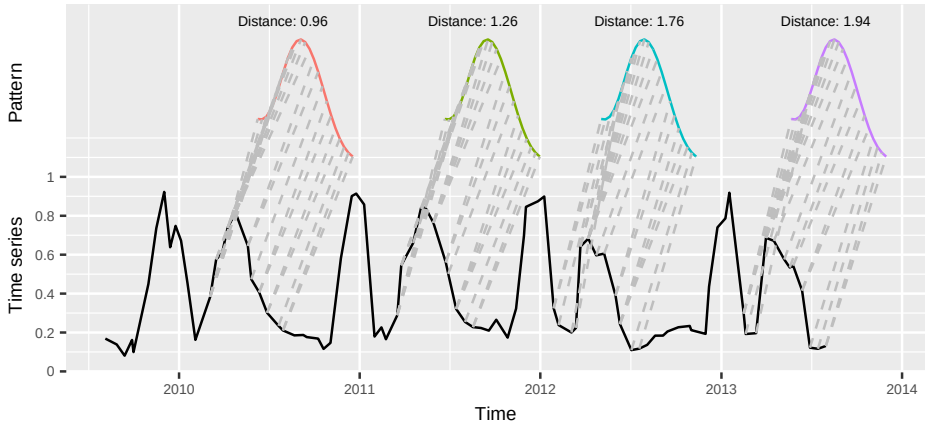

Figure 1: Matches of the known temporal pattern to subintervals of the long-term time series. The solid black line is the long-term time series, the colored lines are the different matches of the same pattern ordered by TWDTW dissimilarity measure, and the gray dashed lines are the matching points.

To use TWDTW for land cover classification, we need the following data sets:

- A set of remote sensing time series for the study area. For example, a tile of a MODIS MOD13Q1 image consists of 4800 x 4800 pixels, covering an area of 10 degrees x 10

degrees at the Equator (Friedl *et al.* 2010). A 15-year (2000-2015) MODIS MOD13Q1 set time series has 23 images per year, with a total of 23 million time series, each with 346 samples.

- A set of time series with land cover information, called *temporal patterns*. Typically, each time series is short and covers one phenological cycle of one land cover type. Examples would be a time series of a soybean crop, or one that describes a mature tropical forest. These temporal patterns can be extracted from the remote sensing image data, if the user knows their spatial and temporal location.

- A set of ground truth points, with spatial and temporal information and land cover classification. These *ground truth* points are used for validation and accuracy assessment.

Based on the information provided by the user about the images to be analysed, our method maps them to a three-dimensional (3-D) array in space-time (Figure 2). This array can have multiple attributes, such as the satellite bands (*e.g.*, "red", "nir", and "blue"), and derived indices (*e.g.*, "NDVI", "EVI", and "EVI2"). This way, each pixel location is associated to a sequence of measurements, building a satellite image time series. Figure 2 shows an example of "EVI" time series for a location in the Brazilian Amazon from 2000 to 2008. In the first two years, the area was covered by forest that was cut in 2002. The area was then used for cattle raising (pasture) for three years, and then for crop production from 2006 to 2008. Satellite image time series are thus useful to describe the dynamics of the landscape and the land use trajectories.

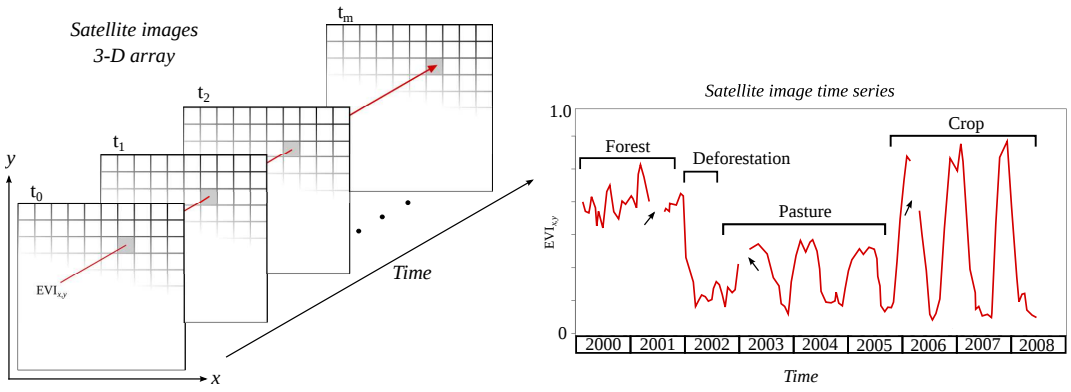

Figure 2: A 3-dimensional array of satellite images (left), an enhanced vegetation index (EVI) time series at the pixel location (x, y) (right). The arrows indicate gaps in the time series. Adapted from Maus *et al.* (2016).

3. dtwSat package overview

dtwSat provides a set of functions for land cover change analysis using satellite image time series. This includes functions to build temporal patterns for land cover types, apply the TWDTW analysis using different weighting functions, visualise the results in a graphical interface, produce land cover maps, and create spatiotemporal plots for land changes. Therefore,

dtwSat gives an end-to-end solution for satellite time series analysis, which users can make a complete land change analysis.

For the **dtwSat** package, the user should provide the following inputs:

- A set of time ordered satellite images, all with the same spatial extent. The user should also inform the date of each image. In R the images should use the `RasterBrick` or `RasterStack` class of the **raster** package.
- A list of temporal patterns, each associated to a time series in **zoo** format.
- A list of known ground truth points, each with spatial and temporal information, in a format readable in R, such as CSV or shapefile.

The **dtwSat** package organizes the data in three `S4` classes of objects: `twdtwTimeSeries`, `twdtwMatches`, and `twdtwRaster`. To store time series we use the class `twdtwTimeSeries`. The objects of class `twdtwTimeSeries` have two slots; the slot called `timeseries` has a list of `zoo` objects; and the slot called `labels` stores the labels of the time series. The class `twdtwMatches` has 3 slots to store inputs and results of the TWDTW analysis. The slots called `timeseries` and `patterns` are objects of the class `twdtwTimeSeries` with the unclassified long-term time series and the temporal patterns, respectively. A third slot called `alignments` has a `list` with detailed information about the matches between the patterns and the unclassified long-term time series. The classes `twdtwTimeSeries` and `twdtwMatches` are used to analyse lists of time series.

The class `twdtwRaster` is used for satellite image time series. This class can store either unclassified raster time series with the satellite raw data, the results of the TWDTW analyis, or a classified raster time series. In both cases, the objects of class `twdtwRaster` have five slots. The slot called `timeseries` is a list of `RasterBrick` or `RasterStack` objects with time ordered satellite images (all with the same temporal and spatial extents); the slot called `timeline` is a vector of class `Date` with dates of the satellite images; the slot called `layers` has the names of satellite bands; the slot called `levels` has levels for the raster values; and the slot called `labels` has labels for the raster values. This class builds upon the R package **raster** to build a multi-attribute 3-D raster in space-time, allowing for multi-band satellite image time series analysis.

4. Classifying a time series

This section describes how to classify one time series, using examples that come with the **dtwSat** package. We will show how to match three temporal patterns ("soybean", "cotton", and "maize") to subintervals of a long-term satellite image time series. These time series have been extracted from a set of MODIS MOD13Q1 (Friedl *et al.* 2010) images and include the vegetation indices "ndvi", "evi", and the original bands "nir", "red", "blue", and "mir". In this example, the classification of crop types for the long-term time series is known.

4.1. Input data

The inputs for the next examples are time series in **zoo** format. The first is an object of class `zoo` with a long-term time series, referred to as `MOD13Q1.ts`, and the second is a `list` of time

series of class `zoo` with the temporal patterns of "soybean", "cotton", and "maize", referred to as `MOD13Q1.patterns.list`.

From `zoo` objects we construct time series of class `twdtwTimeSeries`, for which we have a set of visualization and analysis methods implemented in the **dtwSat** package. The code below builds two objects of class `twdtwTimeSeries`. The first has the long-term time series and second has the temporal patterns. We use the plot method types `timeseries` and `patterns` to shown the objects `ts` in Figure 3 and `MOD13Q1.ts` in Figure 4, respectively. This plot method uses `ggplot` syntax.

```
library(dtwSat)
names(MOD13Q1.patterns.list)
```

```
[1] "Soybean" "Cotton"  "Maize"
```

```
head(MOD13Q1.ts, n = 2)
```

```
             ndvi    evi    red    nir   blue    mir
2009-08-05 0.3169 0.1687 0.1167 0.2250 0.0427 0.2193
2009-08-28 0.2609 0.1385 0.1168 0.1993 0.0548 0.2657
```

```
ts = twdtwTimeSeries(MOD13Q1.ts, labels="Time series")
patterns_ts = twdtwTimeSeries(MOD13Q1.patterns.list)
patterns_ts
```

```
An object of class "twdtwTimeSeries"
Slot "timeseries" length: 3
Slot "labels": [1] "Soybean" "Cotton"  "Maize"
```

```
plot(ts, type = "timeseries") +
  annotate(geom = "text", x = MOD13Q1.ts.labels$from+90, y = 0.98,
  label = MOD13Q1.ts.labels$label, size = 2)
```

```
plot(patterns_ts, type = "patterns")
```

TWDTW uses both amplitude and phase information to classify the phenological cycles in the long-term time series. The differences in the amplitude and phase of the cycles are more clear when we observe the EVI signal in Figures 3 and 4. The EVI peak of the "soybean" time series has a similar amplitude as that of "cotton". However, the "soybean" series peaks in late December while the "cotton" series peaks in early April. The EVI peak of the "maize" time series is at the same period as the peak of "cotton". However, the "maize" time series has smaller amplitude than the "cotton" one. Therefore, combining shape and time information we can improve the time series classification.

4.2. Detection of time series patterns with TWDTW

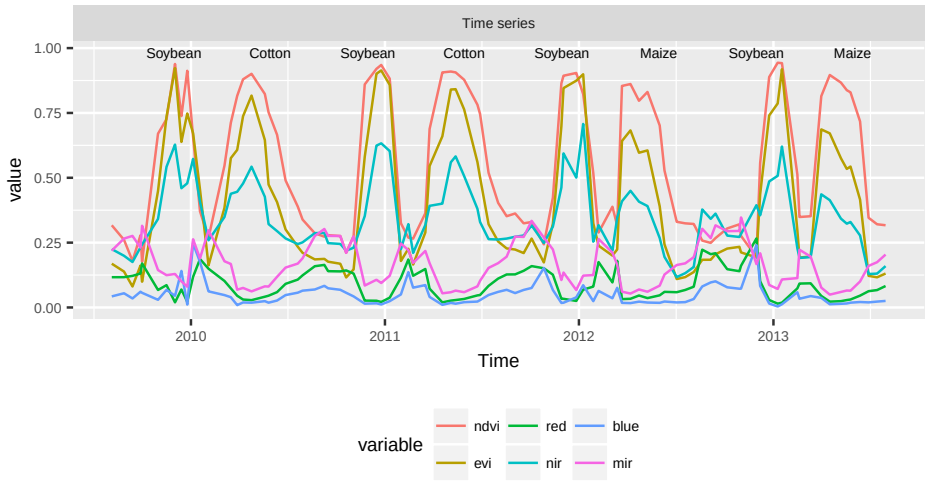

Figure 3: Example of time series based on MODIS product MOD13Q1 (Friedl *et al.* 2010). The labels of the phenological cycle are shown in the plot.

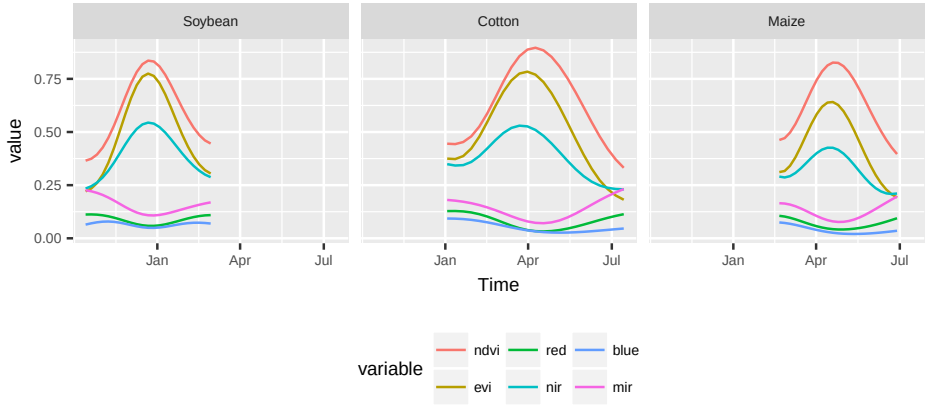

Figure 4: Temporal patterns of soybean, cotton, and maize based on MODIS product MOD13Q1 (Friedl *et al.* 2010).

Each subinterval of the long-term time series in `ts` has a known phenological cycle. We will now compare the known information with the result of the TWDTW algorithm. We use the function `twdtwApply` that returns an R object of class `twdtwMatches` with all matches of each temporal pattern in the time series.

```
log_weight = logisticWeight(alpha = -0.1, beta = 100)
matches =
  twdtwApply(x = ts, y = patterns_ts, weight.fun = log_weight, keep=TRUE)
slotNames(matches)
```

```
[1] "timeseries" "patterns"   "alignments"
```

```
show(matches)
```

```
An object of class "twdtwMatches"
Number of time series: 1
Number of Alignments: 27
Patterns labels: Soybean Cotton Maize
```

To retrieve the complete information of the matches we set `keep=TRUE`. We need this information for the plot methods of the class `twdtwMatches`. The argument `weight.fun` defines the time-weight to the dynamic time warping analysis (Maus *et al.* 2016). By default the time-weight is zero, meaning that the function will run a standard dynamic time warping analysis. The arguments `x` and `y` are objects of class `twdtwTimeSeries` with the unclassified long-term time series and the temporal patterns, respectively. To perform the alignment between the time series the default TWDTW recursion has a symmetric step (for more details and other recursion options see `?stepPattern`). Giorgino (2009) provides a detaild discussion on the recursion steps and other step patterns. For further details and other arguments of the TWDTW analysis see `?twdtwApply`.

In our example we use a logistic weight function for the temporal constraint of the TWDTW algorithm. This function is defined by `logisticWeight`. The **dtwSat** package provides two in-built functions: `linearWeight` and `logisticWeight`. The `linearWeight` function with slope `a` and intercept `b` is given by

$$\omega = a \cdot g(t_1, t_2) + b,$$

and the `logisticWeight` with midpoint `beta`, and steepness `alpha`, given by

$$\omega = \frac{1}{1 + e^{-\alpha(g(t_1,t_2)-\beta)}}.$$

The function g is the absolute difference in days between two dates, t_1 and t_2. The aim of these functions is to control the time warp, e.g. a "large time warp" is needed to match a point of the temporal pattern whose original date is January 1 to a point of the long-term time series whose date is July 1, on the other hand to match January 1 to December 15 has a "small time warp". If there is a large seasonal difference between the pattern and its matching point in time series, an extra cost is added to the DTW distance measure. This constraint controls the time warping and makes the time series alignment dependent on the seasons. This is especially useful for detecting temporary crops and for distinguishing pasture from agriculture. The linear function creates a strong time constraint even for small time differences, including small time warps. The logistic function has a low weight for small time warps and significant costs for bigger time warps, cf. Figure 5. In our previous studies (Maus *et al.* 2016) the logistic-weight had better results than the linear-weight for land cover classification. Users can define different weight functions as temporal constraints in the argument `weight.fun` of the function `twdtwApply`.

4.3. Visualising the result of the TWDTW algorithm

dtwSat provides five ways to visualise objects of class `twdtwMatches` through the plot types: `matches`, `alignments`, `classification`, `path`, and `cost`. The plot type `matches` shows the matching points of the patterns in the long-term time series; the plot type `alignments` shows the alignments and dissimilarity measures; the plot type `path` shows the low cost paths in the

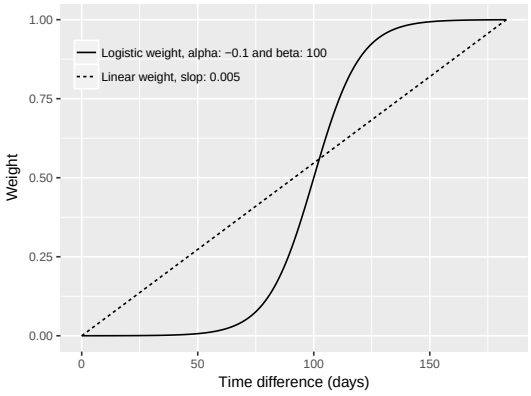

Figure 5: Logistic time-weight function `logisticWeight` with steepness `alpha=-0.1` and midpoint `beta=100`. The x axis shows the absolute difference between two dates in days and the y axis shows the time-weight (Maus *et al.* 2016).

TWDTW cost matrix; and the plot type `cost` allows the visualisation of the cost matrices (local cost, accumulated cost, and time cost); and the plot type `classification` shows the classification of the long-term time series based on the TWDTW analysis. The plot methods for class `twdtwMatches` return a `ggplot` object, so that users can further manipulate the result using the **ggplot2** package. For more details on visualisation functions, please refer to the **dtwSat** documentation in the CRAN (Maus 2015).

We now describe the plot types `matches` and `alignments`. The code bellow shows how to visualise the matching points of the four best matches of "soybean" pattern in the long-term time series, cf. Figure 6.

```
plot(matches, type="matches", patterns.labels="Soybean", k=4)
```

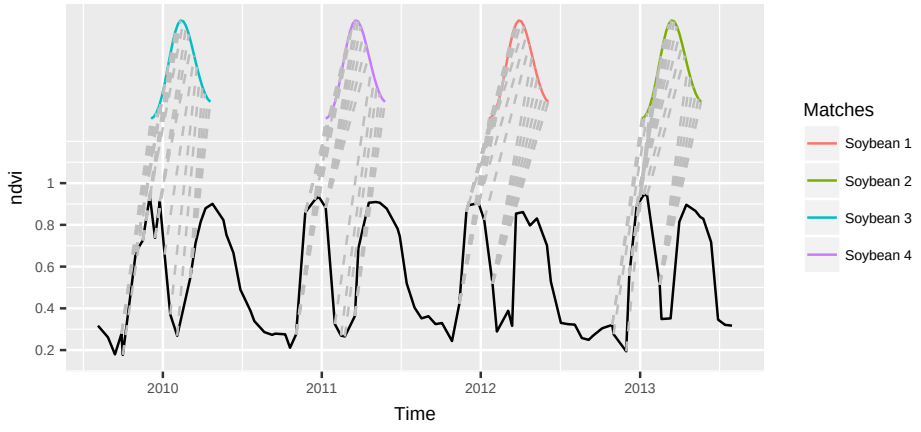

Figure 6: The four best matches of the "soybean" pattern in the time series using a logistic time-weight. The solid black line is the long-term time series; the coloured lines are the temporal patterns; and the grey dashed lines are the respective matching points.

The next example uses the plot type `alignments` to show the alignments of the temporal

patterns (see Figure 7). We set the threshold for the dissimilarity measure to be lower than 3.0. This plot displays the different subintervals of the long-term time series that have alignments whose dissimilarity is less than the specified threshold.

```
plot(matches, type="alignments", attr = "evi", threshold = 3.0)
```

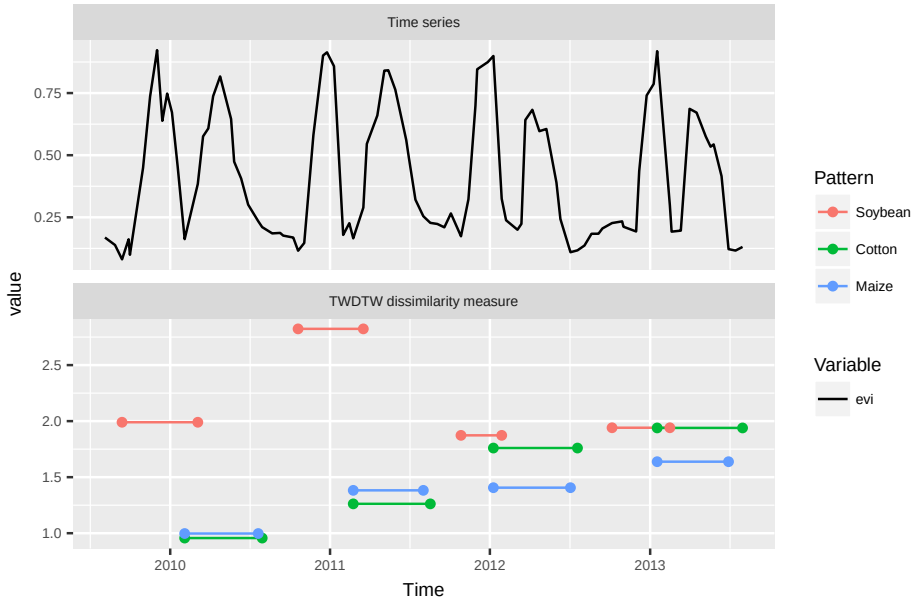

Figure 7: Alignments and dissimilarity measures of the patterns ”soybean”, ”cotton”, and ”maize” to the subintervals of the long-term time series using a logistic time-weight. The solid black line is the EVI time series, and the coloured lines are the alignments of the patterns that have dissimilarity measure lower than three.

Figure 7 shows the alignments of each pattern over the long-term time series, note that we can rank the alignments by their TWDTW dissimilarity, i.e. alignments overlapping the same period usually have distinct dissimilarity, which can be used to rank them. In the figure we can see that maize (blue lines) and cotton (green lines) overlap approximately the same time periods, however, they have distinct dissimilarity measures, and therefore, can be ranked. Observing the time period from January 2010 to July 2010, both soybean, maize, and cotton have at least one overlapping alignment, however in this case the cotton pattern matches better to the interval because its dissimilarity is lower than the others.

4.4. Classifying the long-term time series

Using the matches and their associated dissimilarity measures, we can classify the subintervals of the long-term time series using `twdtwClassify`. To do this, we need to define a period for classification and the minimum overlap between the period and the alignments that intersect with it. For each interval, `twdtwClassify` will select the alignment that has the lowest TWDTW dissimilarity taking into account the minimum overlap condition. For example, in Figure 7 the interval from 1 September 1012 to 28 February 2013 has three overlapping alignments, maize in blue, cotton in green, and soybean in red. Without a minimum overlap the

function `twdtwClassify` would classify this interval as maize, which has the lowest dissimilarity in the period. However, if we set a minimum overlap of 50%, the function `twdtwClassify` classifies the interval as soybean, which is the only class whose alignment overlaps the interval during more than 50% of the time. The interval of classification are usually defined according to the phenological cycles or the agricultural calendar of the region. The classification interval can also be irregular, for details see the argument `breaks` in `?twdtwClassify`

In the example bellow we classify each period of 6 months from September 2009 to September 2013; we set a minimum overlap of 50% between the alignment and the classification period. This means that at least 50% of the alignment has to be contained inside of the classification period. We also use the plot type `classification` to show the classified subintervals of the long-term time series.

```
ts_classification = twdtwClassify(x = matches,
  from = as.Date("2009-09-01"), to = as.Date("2013-09-01"),
  by = "6 month", overlap = 0.5)
plot(ts_classification, type="classification")
```

Figure 8: Classification of each 6 months periods of the time series using results of the TWDTW analysis with logistic time-weight. The solid lines are the attributes of the time series, the background colours indicate the classification of the periods.

By comparing the results of the classified time series in Figure 8 with the crop cycles in Figure 3 we see that the algorithm has classified correctly all the eight subintervals from 2009 to 2013, which are, respectively: “Soybean”, “Cotton”, “Soybean”, “Cotton”, “Soybean”, “Maize”, “Soybean”, “Maize”.

5. Producing a land cover map

In this section we present an application of TWDTW for land cover change analysis using satellite image time series. Our input is a set of images, each covering the same geographical

area at different times. Each pixel location is then associated to an unclassified satellite image time series. We assume to have done field work in the area; for some pixel locations and time periods, we know what is the land cover. We then will show how to obtain a set of template patterns, based on the field samples and how to apply these patterns to land cover classification of the set of images. In the end of this section we show how to perform land cover change analysis and accuracy assessment.

As an example we classify approximately 5300 km^2 in a tropical forest region in Mato Grosso, Brazil (Figure 9). This is a sequence of 160 images with 999 pixels each for 6 years, from 2007 to 2013. We also have a set of 603 ground truth samples of the following classes: "Forest", "Cotton-fallow", "Soybean-cotton", "Soybean-maize", and "Soybean-millet". The satellite images and the field samples used in the examples come with **dtwSat** package.

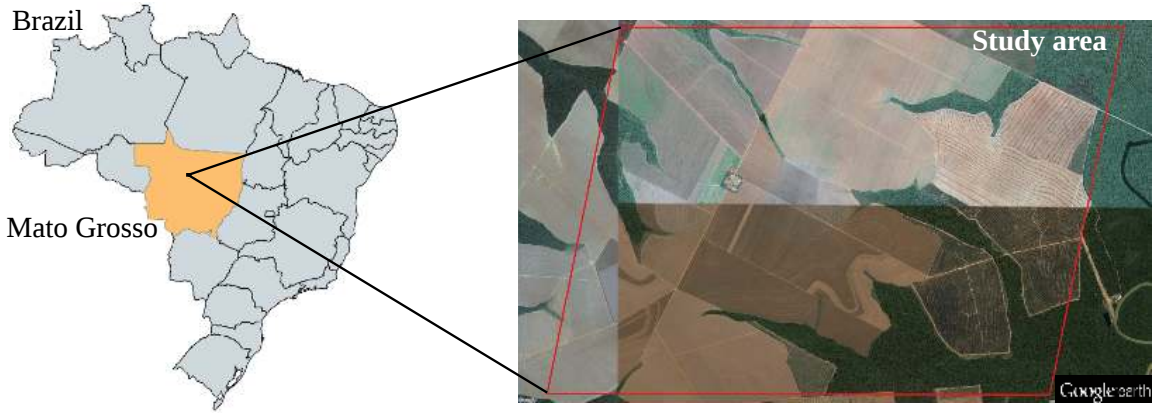

Figure 9: Study area in Mato Grosso, Brazil, shown in a © Google Earth image. The area was originally covered by tropical forest that has been removed for agricultural use.

5.1. Input data

The inputs are: *a)* the satellite images for a given geographical area, organised as a set of georeferenced raster files in GeoTIFF format, each containing all time steps of a spectral band or index; and *b)* a set of ground truth samples. The satellite images are extracted from the MODIS product MOD13Q1 collection 5 (Friedl *et al.* 2010) and include vegetation indices "ndvi", "evi", and original bands "nir", "red", "blue", and "mir". This product has 250 x 250 m spatial resolution and a 16 day maximum-value composite (MVC) for each pixel location (Friedl *et al.* 2010), meaning that one image can have measurements from different dates. For this reason, MOD13Q1 also includes the "day of the year" (doy) of each pixel as a layer, which we use to keep the time series consistent with the measurements.

The data files for the examples that follow are in the **dtwSat** installation folder *lucc_MT/data/*. The *tif* files include the time series of "blue", "red", "nir", "mir", "evi", "ndvi", and "doy" (day of the year); the text file *timeline* has the dates of the satellite images; the CSV file *samples.csv* has the `longitude`, `latitude`, `from`, `to`, and `label` for each field sample; and the text file *samples_projection* contains information about the cartographic projection of the samples, in the format of coordinate reference system used by `sp::CRS`.

```
data_folder = system.file("lucc_MT/data", package = "dtwSat")
dir(data_folder)
```

```
 [1] "blue.tif"           "doy.tif"            "evi.tif"
 [4] "mir.tif"            "ndvi.tif"           "nir.tif"
 [7] "red.tif"            "samples_projection" "samples.csv"
[10] "timeline"
```

We have stored all the time series for each band in one single file. In this way, we can use the function `raster::brick` to read the satellite images. The algorithm also works when the time steps for each band are split in many files. In this case, the user should call the function `raster::stack` with the appropriate parameters. Because of processing performance, we suggest that interested users group their images in bricks and follow the procedures given below.

```
blue = brick(paste(data_folder,"blue.tif", sep = "/"))
red  = brick(paste(data_folder,"red.tif", sep = "/"))
nir  = brick(paste(data_folder,"nir.tif", sep = "/"))
mir  = brick(paste(data_folder,"mir.tif", sep = "/"))
evi  = brick(paste(data_folder,"evi.tif", sep = "/"))
ndvi = brick(paste(data_folder,"ndvi.tif", sep = "/"))
day_of_year  = brick(paste(data_folder,"doy.tif", sep = "/"))
dates = scan(paste(data_folder,"timeline", sep = "/"), what = "dates")
```

We use these data-sets to create a multiple raster time series, which is used in the next sections for the TWDTW analysis. **dtwSat** provides the constructor `twdtwRaster` that builds a multi-band satellite image time series. The inputs of this function are `RasterBrick` objects with the same temporal and spatial extents, and a vector (`timeline`) with the acquisition dates of the images in the format `"YYYY-MM-DD"`. The argument `doy` is combined with `timeline` to get the real date of each pixel, independently from each other. If `doy` is not provided then the dates of the pixels are given by `timeline`, i.e. all pixels in one image will have the same date. Products from other sensors, such as the Sentinels and Landsat, usually have all pixels with same date, therefore the argument `doy` is not needed. This function produces an object of class `twdtwRaster` with the time series of multiple satellite bands.

```
raster_timeseries = twdtwRaster(blue, red, nir, mir, evi, ndvi,
  timeline = dates, doy = day_of_year)
```

Our second input is a set of ground truth samples in the CSV file *samples.csv*, which has a total of 603 samples divided in five classes: 68 “cotton-fallow”, 138 “forest”, 79 “soybean-cotton”, 134 “soybean-maize”, and 184 “soybean-millet”. Reading this CSV file, we get a `data.frame` object, with the spatial location (`latitude` and `longitude`), starting and ending dates (`from` and `to`), and the `label` for each sample.

```
field_samples = read.csv(paste(data_folder,"samples.csv", sep = "/"))
head(field_samples, 5)
```

```
  longitude  latitude       from         to         label
1 -55.98819 -12.03646 2011-09-01 2012-09-01 Cotton-fallow
2 -55.99118 -12.04062 2011-09-01 2012-09-01 Cotton-fallow
3 -55.98606 -12.03646 2011-09-01 2012-09-01 Cotton-fallow
4 -55.98562 -12.03437 2011-09-01 2012-09-01 Cotton-fallow
5 -55.98475 -12.03021 2011-09-01 2012-09-01 Cotton-fallow
```

```
table(field_samples[["label"]])
```

```
 Cotton-fallow         Forest Soybean-cotton  Soybean-maize Soybean-millet
            68            138             79            134            184
```

```
proj_str = scan(paste(data_folder,"samples_projection", sep = "/"),
  what = "character")
proj_str
```

```
[1] "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
```

5.2. Assessing the separability of temporal patterns

The classification is highly dependent on the quality of the temporal patterns. Therefore, it is useful to perform an analysis to assess the separability of the temporal pattern. Ideally, one would need patterns that, when applied to the set of unknown time series, produce consistent results (see the guidelines for single time series analysis in section 4). For this reason, before performing the land cover mapping, we perform a cross validation step. In this way, the users would assess the separability of their patterns before classifying a large data set.

In the next example we use the function `getTimeSeries` to extract the time series of each field sample from our raster time series. The arguments of the function are a set of raster time series, a `data.frame` with spatial and temporal information about the fields samples (as in the object `field_samples` given above), and a `proj4string` with the projection information. The projection should follow the `sp::CRS` format. The result is an object of class `twdtwTimeSeries` with one time series for each field sample.

```
field_samples_ts = getTimeSeries(raster_timeseries,
  y = field_samples, proj4string = proj_str)
field_samples_ts
```

```
An object of class "twdtwTimeSeries"
Slot "timeseries" length: 603
Slot "labels": [1] "Cotton-fallow" "Cotton-fallow" "Cotton-fallow"
```

To perform the cross-validation we use the function `twdtwCrossValidate`. This function splits the sample time series into training and validation sets using stratified sampling with a simple random sampling within each stratum, for details see `?caret::createDataPartition`. The

function uses the training samples to create the temporal patterns and then classifies the remaining validation samples using `twdtwApply`. The results of the classification are used in the accuracy calculation.

A Generalized Additive Model (GAM) [Hastie:1986,Wood:2011] generates the smoothed temporal patterns based on the training samples. We use the GAM because of its flexibility for non-parametric fits, with less rigorous assumptions on the relationship between response and predictor. This potentially provides better fit to satellite data than purely parametric models, due to the data's inter- and intra-annual variability.

In the next example we set the arguments `times=100` and `p=0.1`, which creates 100 different data partitions, each with 10% of the samples for training and 90% for validation. The other arguments of this function are: the logistic weight function with steepness `-0.1` and midpoint `50` to `weight.fun`; the frequency of the temporal patterns to 8 days `freq=8`, and GAM smoothing formula to `formula = y ~ s(x)`, where function `s` sets up a spline model, with `x` the time and `y` a satellite band (for details see `?mgcv::gam` and `?mgcv::s`). The output is an object of class `twdtwCrossValidation` which includes the accuracy for each data partition. The object has two slots, the first called `partitions` has the index of the samples used for training, the second called `accuracy` has overall accuracy, user's accuracy, producer's accuracy, error matrix, and the data used in the calculation, i.e. reference classes, predicted classes, and TWDTW distance measure.

```
set.seed(1)
log_fun = logisticWeight(alpha=-0.1, beta=50)
cross_validation = twdtwCrossValidate(field_samples_ts, times=100, p=0.1,
  freq = 8, formula = y ~ s(x, bs="cc"), weight.fun = log_fun)
```

Figure 10 and Table 1 show the 95% confidence interval of the mean for user's and producer's accuracy derived from the hundred-fold cross-validation analysis. The user's accuracy gives the confidence and the producer's accuracy gives the sensitivity of the method for each class. In our analysis all classes had high user's and producer's accuracy, meaning that TWDTW has high confidence and sensitivity for the classes included in the analysis. The cross-validation results show that if we randomly select 10% of our sampels to create temporal patterns we can get an overall accuracy of at least 97% in the classification of the remaining samples with 95% confidence level.

	User's			Producer's			Overall		
Class	μ	σ	ci*	μ	σ	ci*	μ	σ	ci*
Cotton-fallow	0.96	(0.01)	[0.96-0.96]	1.00	(0.00)	[1.00-1.00]	0.98	(0.02)	[0.97-0.98]
Forest	1.00	(0.00)	[1.00-1.00]	1.00	(0.00)	[1.00-1.00]			
Soybean-cotton	0.99	(0.03)	[0.98-1.00]	0.89	(0.02)	[0.89-0.89]			
Soybean-maize	0.94	(0.05)	[0.93-0.95]	1.00	(0.01)	[1.00-1.00]			
Soybean-millet	1.00	(0.01)	[1.00-1.00]	0.98	(0.05)	[0.97-0.99]			

* 95% confidence interval.

Table 1: User's and producer's accuracy of the TWDTW cross-validation using 100 resampling-with-replacement. The table shows the standard deviation (σ) and the 95% confidence interval (ci) of the mean (μ).'

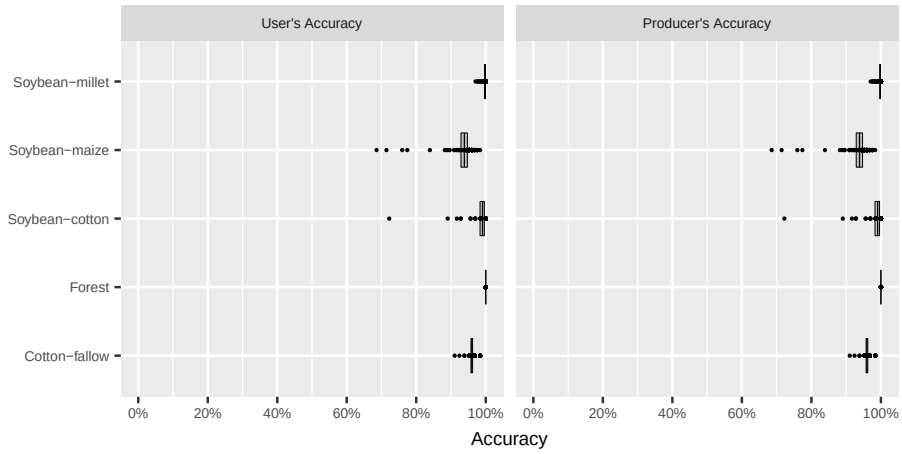

Figure 10: User's and producer's accuracy of the TWDTW cross-validation using 100 resampling-with-replacement. The plot shows the 95% confidence interval of the mean.

5.3. Creating temporal patterns

In the last section we observed that the land cover classes based on our samples are separable using the TWDTW algorithm with high confidence level. Now we randomly select 10% of our samples for training and keep the remaining 90% for validation. The first set of samples are used to create temporal patterns and classify the raster time series, and the second set of samples to assess the final maps. Ideally we would need a second independent set of samples to assess the map, but it would be very difficult to identify different crops without field work. Therefore, we use the same samples used in the cross-validation (subsection 5.2).

```
library(caret)
set.seed(1)
I = unlist(createDataPartition(field_samples[,"label"], p = 0.1))
training_ts = subset(field_samples_ts, I)
validation_samples = field_samples[-I,]
```

We use the function `createPatterns` to produce the temporal patterns based on the training samples. For that, we need to set the desired temporal frequency of the patterns and the smoothing function for the GAM model. In the example below, we set `freq=8` to get temporal patterns with a frequency of 8 days, and the GAM smoothing formula `formula = y ~ s(x)`, such as in subsection 5.2).

```
temporal_patterns =
  createPatterns(training_ts, freq = 8, formula = y ~ s(x))
```

The result of the function `createPatterns` is an object of the class `twdtwTimeSeries`. We use the plot method `type="patterns"` to show the results of the `createPatterns` in Figure 11.

```
plot(temporal_patterns, type = "patterns") +
  theme(legend.position = c(.8,.25))
```

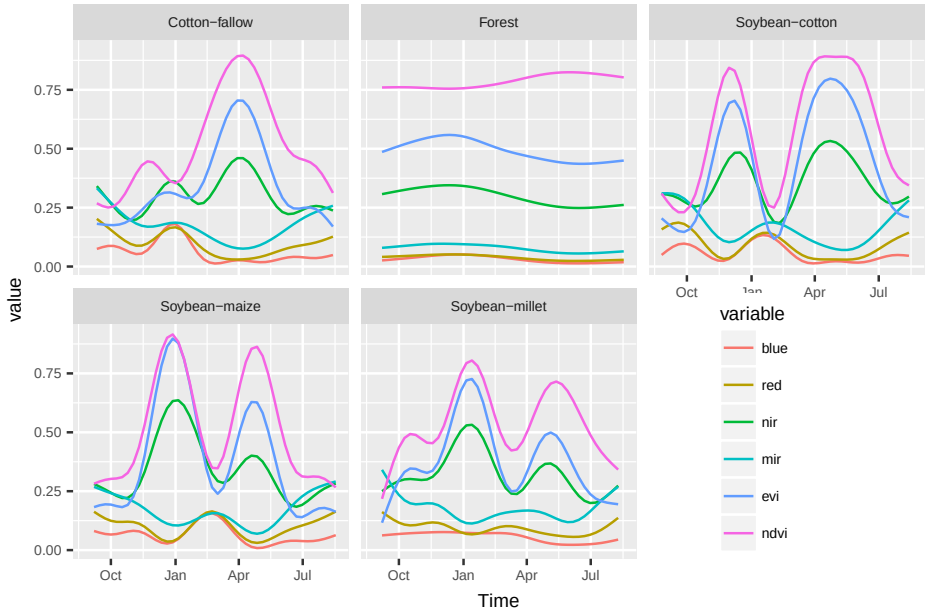

Figure 11: Temporal patterns of Forest, Cotton-fallow, Soybean-cotton, Soybean-maize, and Soybean-millet based on the ground truth samples.

Our method is not restricted to cases where the temporal patterns are obtained from the set of images, such as in the example above. Once can also use patterns from a different set of images or defined in other studies, as long as these temporal patterns stand for the study area and their bands match the bands in the unclassified images.

5.4. Classifying the image time series

After obtaining a consistent set of temporal patterns, we use the function `twdtwApply` to run the TWDTW analysis for each pixel location in the raster time series. The input raster time series in the object `twdtwRaster` should be longer or have approximatly the same length as the temporal patterns. This function retrieves an object of class `twdtwRaster` with the TWDTW dissimilarity measure of the temporal patterns for each time period. The arguments `overwrite` and `format` are passed to `raster::writeRaster`. The arguments `weight.fun` and `overlap` are described in section 4. Here we set the parameters of the time weight (logistic function) base on our the experience about the phenological cycle of the vegetation in the study area. In the next example, we classify the raster time series using the temporal patterns in `temporal_patterns` obtained as described above. The result is a `twdtwRaster` with five layers; each layer contains the TWDTW dissimilarity measure for one temporal pattern over time. We use the plot type `distance` to illustrate the TWDTW dissimilarity for each temporal pattern in 2008, cf. Figure 12.

```
log_fun = logisticWeight(alpha=-0.1, beta=50)
twdtw_dist = twdtwApply(x = raster_timeseries, y = temporal_patterns,
  overlap = 0.5, weight.fun = log_fun, overwrite=TRUE, format="GTiff")
```

```
[1] "Procesing chunk 1/1"
```

```
plot(x = twdtw_dist, type="distance")
```

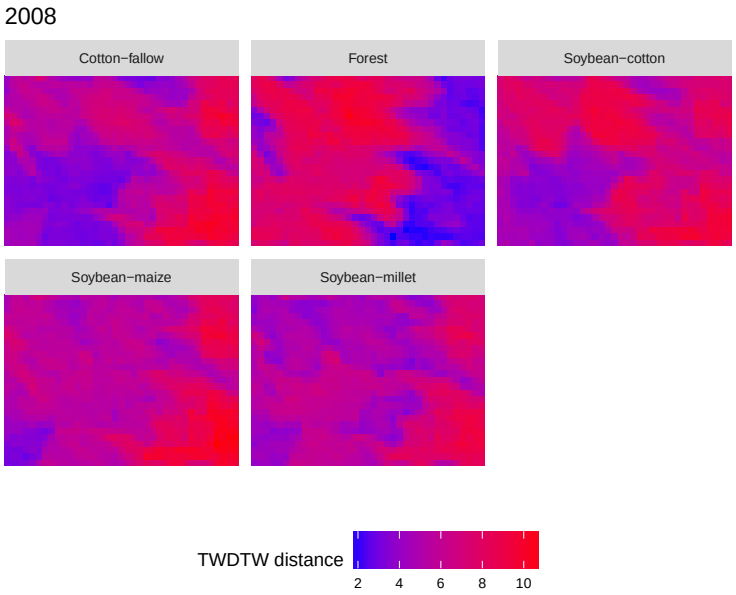

Figure 12: Illustration of the TWDTW dissimilarity from each temporal pattern in 2008. The blue areas are more similar to the pattern and the red areas are less similar to the pattern.

The results of the example above can be used to create categorical land cover maps. The function `twdtwClassify` selects the most similar pattern for each time period and retrieves a `twdtwRaster` object with the time series of land cover maps. The resulting object includes two layers, the first has the classified categorical maps and the second has the TWDTW dissimilarity measure.

```
land_cover_maps = twdtwClassify(twdtw_dist, format="GTiff", overwrite=TRUE)
```

5.5. Looking at the classification results

The classification results can be visualised using the `plot` methods of the class `twdtwRaster`, which supports four plot types: “maps”, “area”, “changes”, and “distance”. The `type="maps"` shows the land cover classification maps for each period, cf. Figure 13.

```
plot(x = land_cover_maps, type = "maps")
```

The next example shows the accumulated area for each class over time, using `type="area"`, cf. Figure 14.

```
plot(x = land_cover_maps, type = "area")
```

Users can also view the land cover transition for each time period, by setting `type="changes"`. For each land cover class and each period, the plot shows gains and losses in area from the other classes. This is the visual equivalent of a land transition matrix, cf. Figure 15.

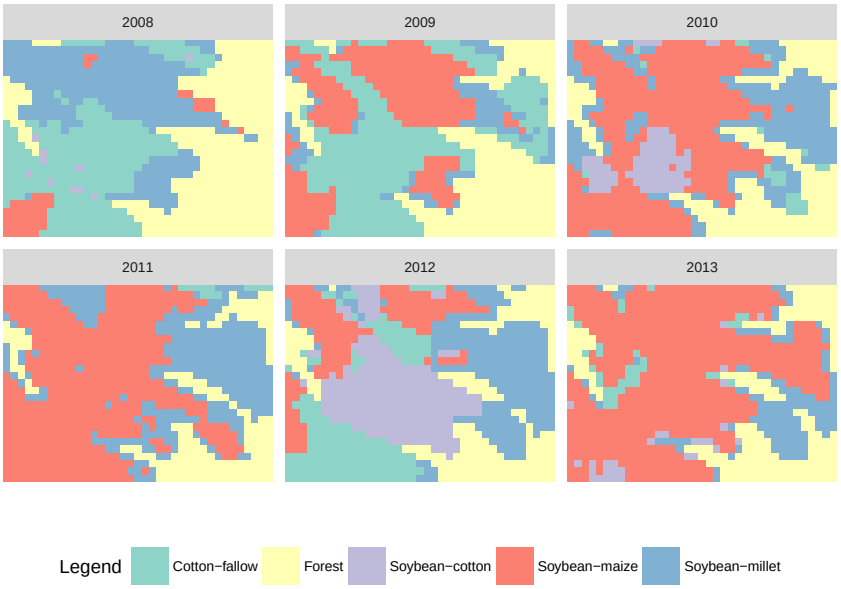

Figure 13: Land cover maps for each year from 2008 to 2013.

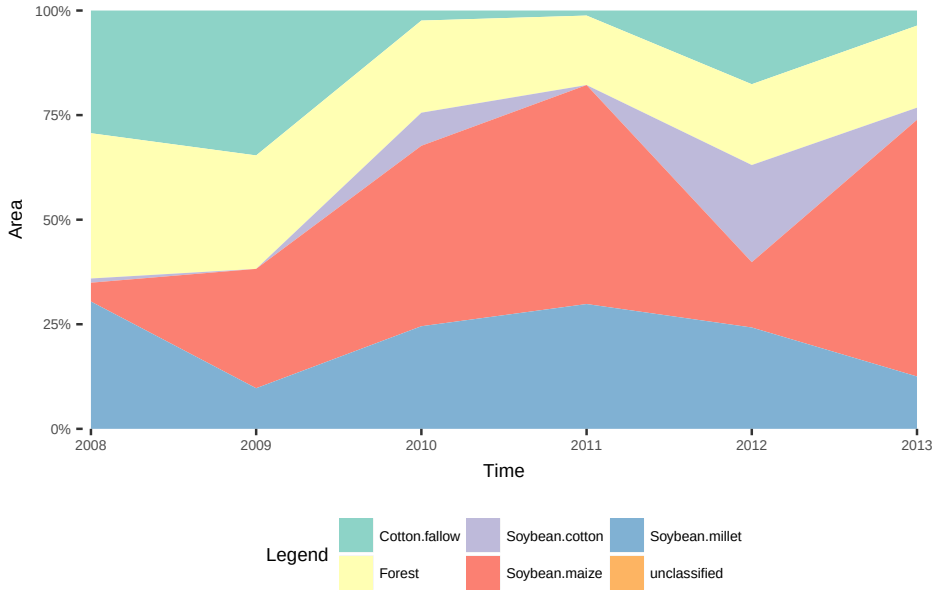

Figure 14: Percentage of area for each land cover class from 2008 to 2013.

```
plot(x = land_cover_maps, type = "changes")
```

We can also look at the dissimilarity of each classified pixel setting `type="distance"`. This plot can give a measure of the uncertainty of the classification of each pixel for each time period, cf. Figure 16.

```
plot(x = land_cover_maps, type="distance")
```

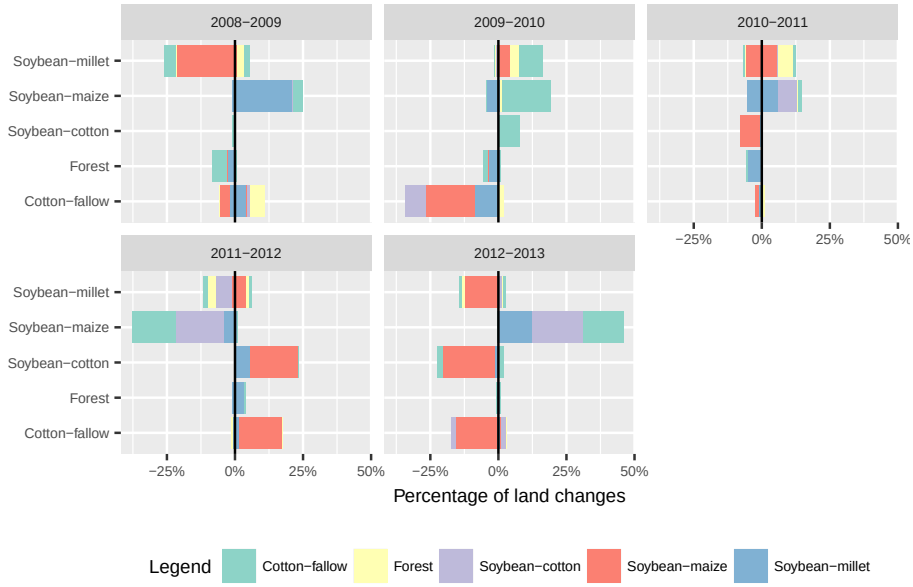

Figure 15: Gains and losses in area from the other classes. The y axis shows the actual class; the positive direction of x axis shows the gains and the negative direction of x axis shows the losses of the classes indicated in y. The colors indicate from/to which classes the gains/losses belong.

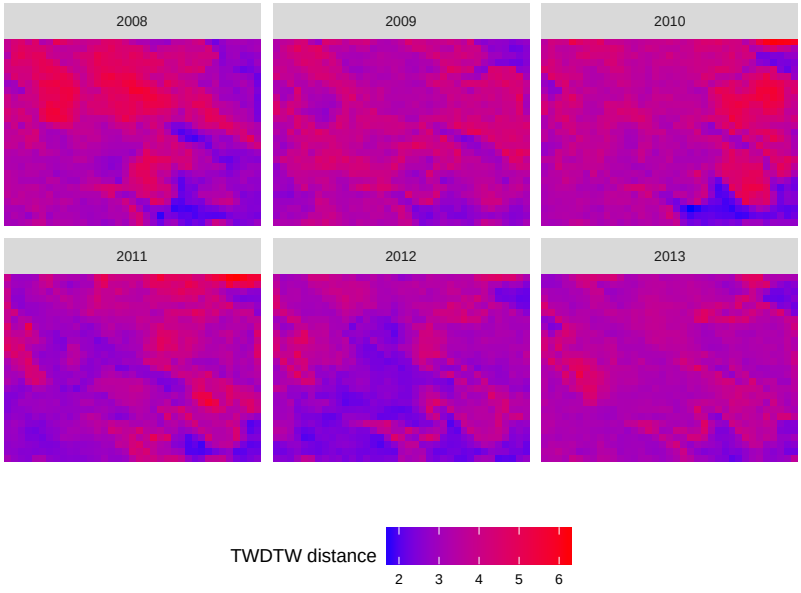

Figure 16: TWDTW dissimilarity measure for each pixel over each classified period. The blue areas have high confidence and the red areas have low confidence in the classification.

5.6. Assessing classification accuracy

In this section we show how to assess the classification. **dtwSat** provides a function called

`twdtwAssess`, which computes a set of accuracy metrics, and adjusted area such as proposed by Olofsson *et al.* (2013) and Olofsson *et al.* (2014). The inputs of this function are the classified map (an object of class `twdtwRaster`), and a set of samples for validation (an object of class `data.frame` or `sp::SpatialPointsDataFrame`). Besides coordinates, the samples should also have starting dates, ending dates, and lables compatible with the labels in the map (for details see subsection 4.1). The output of `twdtwAssess` is an object of class `twdtwAssessment`, which includes four slots: 1) `accuracyByPeriod` is a list of metrics for each time period, including overall accuracy, user's accuracy, produce's accuracy, error matrix (confusion matrix), and adjusted area; 2) `accuracySummary` has the accuracy and adjusted area accumulated over all time periods; 3) `data` is a `SpatialPointsDataFrame` with sample ID, period ID, starting date, ending date, reference label, predicted label, and TWDTW distance; and 4) `map` is a twdtwRaster with the raster maps. The next example uses `twdtwAssess` to compute the accuracy of the maps (`land_cover_maps`) using the validation samples (`validation_samples`) with a 95% confidence level.

```
maps_assessment = twdtwAssess(land_cover_maps, y = validation_samples,
  proj4string = proj_str, conf.int=.95)
```

The results of the assessment in Table 2, 3, and 4 are accumulated over the whole time period, i.e. the total mapped area is equal to the surface area times the number of maps. It is possible to assess and visualise each period independently from each other. However, our samples are irregularly distributed over time and some classes do not have samples in all time period, which limits the analysis of each time period independently from each other.

	Reference class							
Map class	Cotton-fallow	Forest	Soybean-cotton	Soybean-maize	Soybean-millet	Total	Area	w
Cotton-fallow	61	0	3	0	0	64	47600561	0.148
Forest	0	124	0	0	0	124	74754883	0.232
Soybean-cotton	0	0	62	0	0	62	18782634	0.058
Soybean-maize	0	0	6	120	0	126	110173564	0.343
Soybean-millet	0	0	0	0	165	165	70354380	0.219
Total	61	124	71	120	165	541	321666022	1.000

Table 2: Error matrix of the map classification based on TWDTW analysis. The area is in the map unit, in this case m^2. w is the proportion of area mapped for each class.

As we can see in Table 2 only nine samples were misclassified, all of them from the reference class "Soybean-cotton". From these samples six were classified as "Soybean-maize", and three as "Cotton-fallow". As we see in Table 3 the only class with producer's accuracy lower than 100% was "Soybean-cotton", reaching 72% with high uncertainty (±13%). The user's accuracy for all classes was higer than 95%, with maximun uncertainty of ±5%. To visualise the misclassified samples on top of the maps we can use the plot `type="map"`

for objects of class `twdtwAssessment`, such that `plot(x = maps_assessment, type="map", samples="incorrect")`. The user can also set the argument `samples` to see correctly classified samples `samples="correct"`, or to see all samples `samples="all"`.

The Figure 17 shows that the misclassified samples are all in the map from 2012. The six samples of "Soybean-cotton" classified as "Soybean-maize" are within a big area of "Soybean-maize" and the three samples of "Soybean-cotton" classified as "Cotton-fallow" are near the border between this two classes. This errors might be related to the mixture of different classes in the same pixel.

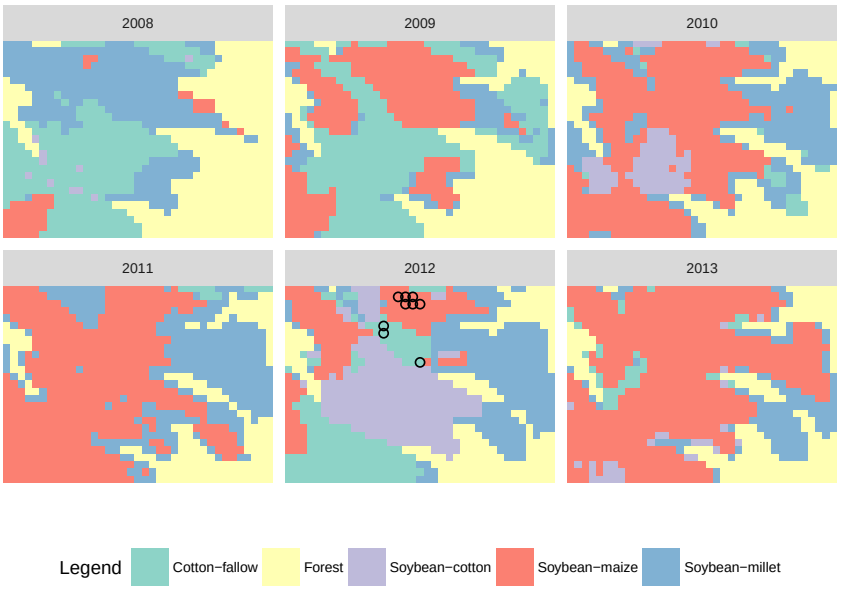

Figure 17: Incorrect classified samples.

	Reference class								
Map class	Cotton-fallow	Forest	Soybean-cotton	Soybean-maize	Soybean-millet	Total	User's*	Producers's*	Overall*
Cotton-fallow	0.14	0.00	0.00	0.00	0.00	0.14	0.95±0.05	1.00±0.00	0.98±0.01
Forest	0.00	0.23	0.00	0.00	0.00	0.23	1.00±0.00	1.00±0.00	
Soybean-cotton	0.01	0.00	0.06	0.02	0.00	0.08	1.00±0.00	0.72±0.13	
Soybean-maize	0.00	0.00	0.00	0.33	0.00	0.33	0.95±0.04	1.00±0.00	
Soybean-millet	0.00	0.00	0.00	0.00	0.22	0.22	1.00±0.00	1.00±0.00	
Total	0.15	0.23	0.06	0.34	0.22	1.00			

* 95% confidence interval.

Table 3: Accuracy and error matrix in proportion of area of the classified map.

In Table 4 we can see the mapped and the adjusted area. This is the accumulated area over the

whole period, i.e. the sum of all maps from 2008 to 2013. As the "Forest" and "Soybean-millet" did not have omission (100% producer's accuracy) or comission (100% user's accuracy) erros, we immediately see that their mapped area is equal to their adjusted area (Table 4). To help the analysis of the other classes we use the plot `type="area"` for class `twdtwAssessment`, such that `plot(x = maps_assessment, type="area", perc=FALSE)`. Figure 18 shows the accumulated area mapped and adjusted for all classes. In this figure we see that our method overestimated the area of "Soybean-maize", i.e. the mapped area (110173564 m^2) is bigger than the adjusted area (104927204 m^2) plus the confidence interval 4113071 m^2. Meanwhile we underestimated the area of "Soybean-cotton", i.e. its mapped area (18782634 m^2) is smaller than the adjusted area (26260270 m^2) plus the confidence interval (4805205 m^2). The mapped area of "Cotton-fallow" (47600561 m^2) is within the confidence interval of the adjusted area (45369285 ± 2484480 m^2). To improve the accuracy assessment and area estimations the field samples should be better distributed over time, which would also allow for better land cover changes assessment.

Class	Mapped area	Adjusted area	Margin of error*
Cotton-fallow	47600561	45369285	±2484480
Forest	74754883	74754883	±0
Soybean-cotton	18782634	26260270	±4805205
Soybean-maize	110173564	104927204	±4113071
Soybean-millet	70354380	70354380	±0

* 95% confidence interval.

Table 4: Mapped and adjusted, accumulated over the whole period, i.e. the sum from the sum of the maps from 2008 to 2013. The area is in the map unit, in this case m^2.

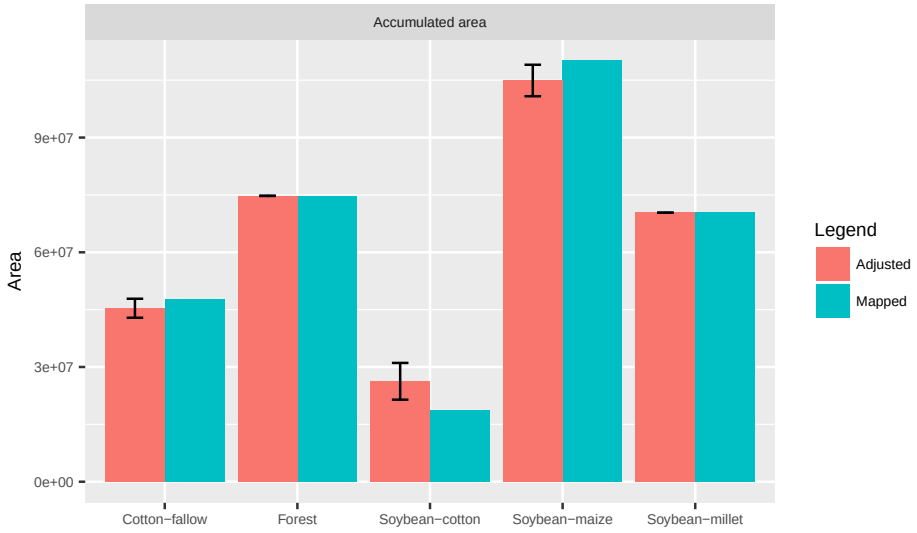

Figure 18: Mapped and adjusted, accumulated over the whole period, i.e. the sum from the sum of the maps from 2008 to 2013. The area is in the map unit, in this case m^2.

6. Conclusions and Discussion

The overall accuracy of the classification with a 95% confidence level is within 97% and 99%. With same confidence level, user's and producer's accuracy are between 90% and 100% for all classes, except for "Soybean-cotton", which has wide confidence interval for user's accuracy, between 59% and 85%. A small sample size will likely have large confidence intervals (Foody 2009), therefore, this analysis could be improved by increasing the number of "Soybean-cotton" samples. In addition, our access to field information is limited to a set of samples irregularly distributed over time, which are not enough to assess each mapped period independently from each other. Nevertheless, the results of the accuracy assessment show that the TWDTW has great potential to classify different crop types.

DTW based approaches have achieved good results for land cover classification (Petitjean *et al.* 2012; Maus *et al.* 2016), however, a reduced number of points in the time series will negatively impact the accuracy. Remotely sensed images often present noise and poor coverage due to clouds, aerosol load, surface directional effects, and sensor problems. This leads to large amount of gaps in satellite image time series. Therefore, methods that deal with irregular temporal sampling, i.e. irregular sampling intervals, have great potential to fully exploit the available satellite images archive. DTW is known to be one of the most robust methods for irregular time series (Keogh and Ratanamahatana 2005; Tormene *et al.* 2009). It was successfully applied for satellite time series clustering using FORMOSAT-2 (Petitjean *et al.* 2012) and using MODIS (Maus *et al.* 2016). Petitjean *et al.* (2012), for example, showed that clustering based on DTW is consistent even when there are several images missing per year because of cloud cover. However, the effect of a reduced number of samples in the time needs to be better evaluated in order to point out the limiting gap size for satellite image time series analysis using DTW based methods.

The DTW approaches will search for the matches of a temporal pattern, therefore the number of points in the time series should represent the phenological cycles of different land cover types. The number of available observations might be a limitation for sensors with lower temporal resolution, such as Landsat. We believe that this limitation could be addressed, for example, by combining TWDTW analysis with pixel-based compositing techniques (Griffiths *et al.* 2013; White *et al.* 2014). These approaches have become more popular with the opening of the USGS Landsat archive and could be used to increase the availability of gap-free time series (Gómez *et al.* 2016).

dtwSat provides a dissimilarity measure in the n-dimensional space, allowing multispectral satellite image time series analysis. Our experience using MODIS data sets has shown that n-dimensional analysis, i.e. using RED, NIR, NDVI, EVI, and NDVI, increases the separability among classes when compared to single band analysis, for example using only EVI or NDVI. Further studies on multispectral data using TWDTW analysis will help to optimize the selection of bands.

The main goal of **dtwSat** package is to make TWDTW accessible for researchers. The package supports the full cycle of land cover classification using image time series, ranging from selecting temporal patterns to visualising and assessing the results. The current version of the **dtwSat** package provides a pixel-based time series classification method. We envisage that future versions of the package could include local neighborhoods to reduce border effects and improve classification homogeneity.

The **dtwSat** package provides two in-built functions for linear and logistic time weight. In the

current version of the package the parameters of the weight functions are set manually to the same value for all land cover classes. Future versions could include methods to search for the best parameters for each land cover class using field data.

Nowadays, there are large open archives of Earth Observation data, but few open source methods for analysing them. With this motivation, this paper provides guidance on how to use the Time-Weighed Dynamic Time Warping (TWDTW) method for remote sensing applications. As we have discussed in a companion paper (Maus *et al.* 2016), the TWDTW method is well suited for land cover change analysis of satellite image time series.

The TWDTW algorithm is computationally intensive and for large areas one should consider parallel processing. The algorithm is pixel time series based, i.e. each time series is processed independently from each other, and therefore, it can be easily parallelized. To aim for maximum usage by the scientific community, the **dtwSat** package described in this paper works with well-known R data classes such as provided by **raster**, which offers the option to work with raster data sets stored on disk that are too large to be loaded into memory (RAM) at once (Hijmans 2015). We are planning improvements, so that **dtwSat** can also be combined with array databases, such as SciDB (Stonebraker *et al.* 2013). We believe that combining array databases with image time series analysis software such as presented here is one way forward to scaling the process of information extracting from very large Earth Observation data.

Acknowledgments

Victor Maus has been supported by the Institute for Geoinformatics, University of Münster (Germany), and by the Earth System Science Center, National Institute for Space Research (Brazil). Gilberto Camara's term as Brazil Chair at IFGI has been supported by CAPES (grant 23038.007569/2012-16). Gilberto's work is also supported by FAPESP e-science program (grant 2014-08398-6) and CNPq (grant 312151/2014-4).

References

Berndt DJ, Clifford J (1994). "Using Dynamic Time Warping to Find Patterns in Time Series." In UM Fayyad, R Uthurusamy (eds.), *KDD Workshop*, pp. 359–370. AAAI Press. ISBN 0-929280-73-3.

Bivand R, Lewin-Koh N (2015). ***maptools**: Tools for Reading and Handling Spatial Objects*. R package version 0.8-37, URL http://CRAN.R-project.org/package=maptools.

Bivand RS, Pebesma E, Gomez-Rubio V (2013). *Applied Spatial Data Analysis With R, Second edition*. Springer-Verlag, New York. URL http://www.asdar-book.org/.

DeVries B, Verbesselt J, Kooistra L, Herold M (2015). "Robust Monitoring of Small-Scale Forest Disturbances in a Tropical Montane Forest Using Landsat Time Series." *Remote Sensing of Environment*, **161**(0), 107 – 121. doi:10.1016/j.rse.2015.02.012.

Dutrieux L, DeVries B (2014). "**bfastSpatial**: Set of Utilities and Wrappers to Perform Change Detection on Satellite Image Time-Series." doi:10.5281/zenodo.49693. URL https://github.com/dutri001/bfastSpatial.

Foody GM (2009). "Classification accuracy comparison: Hypothesis tests and the use of confidence intervals in evaluations of difference, equivalence and non-inferiority." *Remote Sensing of Environment*, **113**(8), 1658 – 1663. ISSN 0034-4257. doi:10.1016/j.rse.2009.03.014.

Friedl MA, Sulla-Menashe D, Tan B, Schneider A, Ramankutty N, Sibley A, Huang X (2010). "MODIS Collection 5 Global Land Cover: Algorithm Refinements and Characterization of New Datasets." *Remote Sensing of Environment*, **114**(1), 168 – 182. ISSN 0034-4257. doi:10.1016/j.rse.2009.08.016.

Fritz S, See L, You L, Justice C, Becker-Reshef I, Bydekerke L, Cumani R, Defourny P, Erb K, Foley J, Gilliams S, Gong P, Hansen M, Hertel T, Herold M, Herrero M, Kayitakire F, Latham J, Leo O, McCallum I, Obersteiner M, Ramankutty N, Rocha J, Tang H, Thornton P, Vancutsem C, van der Velde M, Wood S, Woodcock C (2013). "The Need for Improved Maps of Global Cropland." *Eos, Transactions American Geophysical Union*, **94**(3), 31–32. ISSN 2324-9250. doi:10.1002/2013EO030006.

Galford GL, Mustard JF, Melillo J, Gendrin A, Cerri CC, Cerri CE (2008). "Wavelet Analysis of MODIS Time Series to Detect Expansion and Intensification of Row-Crop Agriculture in Brazil." *Remote Sensing of Environment*, **112**(2), 576–587.

Giorgino T (2009). "Computing and Visualizing Dynamic Time Warping Alignments in R: The **dtw** Package." *Journal of Statistical Software*, **31**(7), 1–24. doi:10.18637/jss.v031.i07.

Gómez C, White JC, Wulder MA (2016). "Optical remotely sensed time series data for land cover classification: A review." *ISPRS Journal of Photogrammetry and Remote Sensing*, **116**, 55 – 72. ISSN 0924-2716. doi:10.1016/j.isprsjprs.2016.03.008.

Goslee S (2011). "Analyzing Remote Sensing Data in R: The landsat Package." *Journal of Statistical Software*, **43**(1), 1–25. ISSN 1548-7660. doi:10.18637/jss.v043.i04.

Griffiths P, van der Linden S, Kuemmerle T, Hostert P (2013). "A Pixel-Based Landsat Compositing Algorithm for Large Area Land Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **6**(5), 2088–2101. ISSN 1939-1404. doi:10.1109/JSTARS.2012.2228167.

Hijmans RJ (2015). ***raster**: Geographic Data Analysis and Modeling*. R package version 2.5-2, URL http://CRAN.R-project.org/package=raster.

Jönsson P, Eklundh L (2002). "Seasonality Extraction by Function Fitting to Time-Series of Satellite Sensor Data." *IEEE Transactions on Geoscience and Remote Sensing*, **40**(8), 1824–1832. doi:10.1109/TGRS.2002.802519.

Jönsson P, Eklundh L (2004). "TIMESAT – A Program for Analyzing Time-Series of Satellite Sensor Data." *Computers & Geosciences*, **30**(8), 833 – 845. ISSN 0098-3004. doi:10.1016/j.cageo.2004.05.006.

Kennedy RE, Yang Z, Cohen WB (2010). "Detecting Trends in Forest Disturbance and Recovery Using Yearly Landsat Time Series: 1. LandTrendr – Temporal Segmentation Algorithms." *Remote Sensing of Environment*, **114**(12), 2897–2910. ISSN 0034-4257. doi:10.1016/j.rse.2010.07.008.

Keogh E, Ratanamahatana CA (2005). "Exact Indexing of Dynamic Time Warping." *Knowledge Information Systems*, **7**(3), 358–386.

Kuhn M, with contributions from Jed Wing, Weston S, Williams A, Keefer C, Engelhardt A, Cooper T, Mayer Z, Kenkel B, the R Core Team, Benesty M, Lescarbeau R, Ziem A, Scrucca L, Tang Y, Candan C (2016). ***caret**: Classification and Regression Training*. R package version 6.0-64, URL http://CRAN.R-project.org/package=caret.

Lambin E, Linderman M (2006). "Time Series of Remote Sensing Data for Land Change Science." *IEEE Transactions on Geoscience and Remote Sensing*, **44**(7), 1926–1928. ISSN 0196-2892. doi:10.1109/TGRS.2006.872932.

le Maire G, Dupuy S, Nouvellon Y, Loos RA, Hakamada R (2014). "Mapping Short-Rotation Plantations at Regional Scale Using MODIS Time Series: Case of Eucalypt Plantations in Brazil." *Remote Sensing of Environment*, **152**(0), 136 – 149. doi:10.1016/j.rse.2014.05.015.

Lunetta RS, Knight JF, Ediriwickrema J, Lyon JG, Worthy LD (2006). "Land-cover Change Detection Using Multi-Temporal MODIS NDVI Data." *Remote Sensing of Environment*, **105**(2), 142 – 154. doi:10.1016/j.rse.2006.06.018.

Maus V (2015). ***dtwSat**: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis*. R package version 0.1.0, URL http://CRAN.R-project.org/package=dtwSat.

Maus V, Camara G, Cartaxo R, Sanchez A, Ramos FM, de Queiroz GR (2016). "A Time-Weighted Dynamic Time Warping Method for Land-Use and Land-Cover Mapping." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **9**(8), 3729–3739. doi:10.1109/JSTARS.2016.2517118.

Meyer D, Buchta C (2015). ***proxy**: Distance and Similarity Measures*. R package version 0.4-15, URL http://CRAN.R-project.org/package=proxy.

Moulds S, Buytaert W, Mijic A (2015). "An Open and Extensible Framework for Spatially Explicit Land Use Change Modelling: The LULCC R Package." *Geoscientific Model Development*, **8**(10), 3215–3229. doi:10.5194/gmd-8-3215-2015.

Müller M (2007). *Information Retrieval for Music and Motion*. Springer-Verlag, London.

Olofsson P, Foody GM, Herold M, Stehman SV, Woodcock CE, Wulder MA (2014). "Good practices for estimating area and assessing accuracy of land change." *Remote Sensing of Environment*, **148**, 42 – 57. ISSN 0034-4257. doi:10.1016/j.rse.2014.02.015.

Olofsson P, Foody GM, Stehman SV, Woodcock CE (2013). "Making better use of accuracy data in land change studies: Estimating accuracy and area and quantifying uncertainty using stratified estimation." *Remote Sensing of Environment*, **129**, 122 – 131. doi:10.1016/j.rse.2012.10.031.

Pebesma E (2012). "**spacetime**: Spatio-Temporal Data in R." *Journal of Statistical Software*, **51**(1), 1–30. ISSN 1548-7660. doi:10.18637/jss.v051.i07.

Pebesma EJ, Bivand RS (2005). "Classes and methods for spatial data in **R**." *R News*, **5**(2), 9–13. URL http://CRAN.R-project.org/doc/Rnews/.

Petitjean F, Inglada J, Gancarski P (2012). "Satellite Image Time Series Analysis Under Time Warping." *IEEE Transactions on Geoscience and Remote Sensing*, **50**(8), 3081–3095. ISSN 0196-2892. doi:10.1109/TGRS.2011.2179050.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rabiner L, Juang BH (1993). *Fundamentals of speech recognition*. Prentice-Hall International, Inc., New Jersey.

Reed BC, Brown JF, VanderZee D, Loveland TR, Merchant JW, Ohlen DO (1994). "Measuring Phenological Variability from Satellite Imagery." *Journal of Vegetation Science*, **5**(5), 703–714. doi:10.2307/3235884.

Sakamoto T, Van PC, Kotera, Nguyen KD, Yokozawa M (2009). "Analysis of Rapid Expansion of Inland Aquaculture and Triple Rice-Cropping Areas in a Coastal Area of the Vietnamese Mekong Delta Using MODIS Time-Series Imagery." *Landscape and Urban Planning*, **92**(1), 34–46. doi:10.1016/j.landurbplan.2009.02.002.

Sakoe H, Chiba S (1971). "A Dynamic Programming Approach to Continuous Speech Recognition." In *Proceedings of the Seventh International Congress on Acoustics, Budapest*, volume 3, pp. 65–69. Akadémiai Kiadó, Budapest.

Sakoe H, Chiba S (1978). "Dynamic Programming Algorithm Optimization for Spoken Word Recognition." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **26**(1), 43–49. doi:10.1109/TASSP.1978.1163055.

Stonebraker M, Brown P, Zhang D, Becla J (2013). "SciDB: A Database Management System for Applications with Complex Analytics." *Computing in Science & Engineering*, **15**(3), 54–62.

Tormene P, Giorgino T, Quaglini S, Stefanelli M (2009). "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation." *Artificial Intelligence in Medicine*, **45**(1), 11 – 34. doi:10.1016/j.artmed.2008.11.007.

Tuck SL, Phillips HR, Hintzen RE, Scharlemann JP, Purvis A, Hudson LN (2014). "**MODISTools** – Downloading and Processing MODIS Remotely Sensed Data in R." *Ecology and Evolution*, **4**(24), 4658–4668. ISSN 2045-7758. doi:10.1002/ece3.1273.

Velichko V, Zagoruyko N (1970). "Automatic Recognition of 200 Words." *International Journal of Man-Machine Studies*, **2**(3), 223–234. ISSN 0020-7373. doi:10.1016/S0020-7373(70)80008-6.

Verbesselt J, Hyndman R, Newnham G, Culvenor D (2010a). "Detecting Trend and Seasonal Changes in Satellite Image Time Series." *Remote Sensing of Environment*, **114**(1), 106–115. ISSN 0034-4257. doi:10.1016/j.rse.2009.08.014.

Verbesselt J, Hyndman R, Zeileis A, Culvenor D (2010b). "Phenological Change Detection While Accounting for Abrupt and Gradual Trends in Satellite Image Time Series." *Remote Sensing of Environment*, **114**(12), 2970 – 2980. ISSN 0034-4257. doi:10.1016/j.rse.2010.08.003.

Verbesselt J, Zeileis A, Herold M (2011). "Near Real-Time Disturbance Detection in Terrestrial Ecosystems Using Satellite Image Time Series: Drought Detection in Somalia." *Working papers*, Faculty of Economics and Statistics, University of Innsbruck.

Verbesselt J, Zeileis A, Herold M (2012). "Near Real-Time Disturbance Detection Using Satellite Image Time Series." *Remote Sensing of Environment*, **123**(0), 98 – 108. doi:10.1016/j.rse.2012.02.022.

Wardlow BD, Egbert SL, Kastens JH (2007). "Analysis of Time-Series MODIS 250 m Vegetation Index Data for Crop Classification in the U.S. Central Great Plains." *Remote Sensing of Environment*, **108**(3), 290 – 310. doi:10.1016/j.rse.2006.11.021.

White JC, Wulder MA, Hobart GW, Luther JE, Hermosilla T, Griffiths P, Coops NC, Hall RJ, Hostert P, Dyk A, Guindon L (2014). "Pixel-Based Image Compositing for Large-Area Dense Time Series Applications and Science." *Canadian Journal of Remote Sensing*, **40**(3), 192–212. doi:10.1080/07038992.2014.945827.

Wickham H (2009). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag, New York. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book.

Wood S (2006). *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC.

Wood SN (2000). "Modelling and Smoothing Parameter Estimation with Multiple Quadratic Penalties." *Journal of the Royal Statistical Society (B)*, **62**(2), 413–428.

Wood SN (2003). "Thin-Plate Regression Splines." *Journal of the Royal Statistical Society (B)*, **65**(1), 95–114.

Wood SN (2004). "Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models." *Journal of the American Statistical Association*, **99**(467), 673–686.

Wood SN (2011). "Fast Stable Restricted Maximum Likelihood and Marginal Likelihood Estimation of Semiparametric Generalized Linear Models." *Journal of the Royal Statistical Society (B)*, **73**(1), 3–36.

Xiao X, Boles S, Liu J, Zhuang D, Frolking S, Li C, Salas W, III BM (2005). "Mapping Paddy Rice Agriculture in Southern China Using Multi-Temporal MODIS Images." *Remote Sensing of Environment*, **95**(4), 480 – 492. doi:10.1016/j.rse.2004.12.009.

Zeileis A, Grothendieck G (2005). "**zoo**: S3 Infrastructure for Regular and Irregular Time Series." *Journal of Statistical Software*, **14**(6), 1–27. URL http://www.jstatsoft.org/v14/i06/.

Zhang X, Friedl MA, Schaaf CB, Strahler AH, Hodges JC, Gao F, Reed BC, Huete A (2003). "Monitoring Vegetation Phenology Using MODIS." *Remote Sensing of Environment*, **84**(3), 471 – 475. doi:10.1016/S0034-4257(02)00135-9.

Zhu Z, Woodcock CE, Olofsson P (2012). "Continuous Monitoring of Forest Disturbance Using all Available Landsat Imagery." *Remote Sensing of Environment*, **122**(0), 75–91. ISSN 0034-4257. doi:10.1016/j.rse.2011.10.030. Landsat Legacy Special Issue.

Affiliation:

Victor Maus
Ecosystems Services and Management Program
International Institute for Applied Systems Analysis
Schlossplatz 1
A-2361 Laxenburg, Austria
E-mail: vwmaus1@gmail.com
URL: www.iiasa.ac.at/staff/maus