SITS: Data Analysis and Machine Learning using Satellite Image Time Series

Rolf Simoes *National Institute for Space Research (INPE), Brazil*
Gilberto Camara *National Institute for Space Research (INPE), Brazil*
Alexandre Carvalho *Institute for Applied Economics Research (IPEA), Brazil*
Victor Maus *International Institute for Applied System Analysis (IIASA), Austria*
Gilberto Queiroz *National Institute for Space Research (INPE), Brazil*

Using time series derived from big Earth Observation data sets is one of the leading research trends in Land Use Science and Remote Sensing. One of the more promising uses of satellite time series is its application for classification of land use and land cover, since our growing demand for natural resources has caused major environmental impacts. Here, we present the open source R package for satellite image time series analysis, the `sits` package. The `sits` provides support on how to use statistical learning techniques with image time series. These methods include linear and quadratic discrimination analysis, support vector machines, random forests and neural networks.

Introduction

Earth observation satellites provide a continuous and consistent set of information about the Earth's land and oceans. Most space agencies have adopted an open data policy, making unprecedented amounts of satellite data available for research and operational use. This data deluge has brought about a major challenge: *How to design and build technologies that allow the Earth observation community to analyse big data sets?*

The approach taken in the current work is to develop data analysis methods that work with satellite image time series. The time series are obtained by taking calibrated and comparable measures of the same location in Earth at different times. These measures can be obtained by a single sensor (*e.g.*, MODIS) or by combining different sensors (*e.g.*, Landsat 8 and Sentinel-2). If obtained by frequent revisits, the temporal resolution of these data sets can capture the most important land use changes.

Time series of remote sensing data show that land cover changes do not always occur in a progressive and gradual way, but they may also show periods of rapid and abrupt change followed either by a quick recovery [Lambin et al., 2003]. Analyses of multiyear time series of land surface attributes, their fine-scale spatial pattern, and their seasonal evolution leads to a broader view of land-cover change. Satellite image time series have already been applied to applications such as mapping for detecting forest disturbance [Kennedy et al., 2010], ecology dynamics [Pasquarella et al., 2016], agricultural intensification [Galford et al., 2008] and its impacts on deforestation [Arvor et al., 2012].

In this paper, we present an open source R package for satellite image time series analysis `sits`. The `sits` package provides support on how to use statistical learning techniques with image time series. In a broad sense, statistical learning refers to a class of algorithms for classification and regression analysis [Hastie et al., 2009]. These methods include linear and quadratic discrimination analysis, support vector machines, random forests and neural networks. In a typical classification problem, we have measures that capture class attributes. Based on these measures, referred as training data, one's task is to select a predictive model that allows inferring classes of a larger data set.

In what follows, we describe the main characteristics of the `sits`. The first part describes the basic data structures used in it and the tools used for visualisation and data exploration. Then we show how to do data acquisition from external sources, with an emphasis on the WTSS (an acronym for Web Time Series Service) [Queiroz et al., 2015]. The next sections describe filtering and clustering techniques. We then discuss machine learning techniques for satellite image time series data and how to apply them to image time series. Finally, we present validation methods.

Data Handling and Visualisation Basics in `sits`

The basic data unit in the `sits` package is the "`sits` tibble", which is a way of organizing a set of time series data with associated spatial information. In R, a `tibble` differs from the traditional data frame, insofar as a `tibble` can contain lists embedded as column arguments. Tibbles are part of the `tidyverse`, a collection of R package designed to work together in data manipulation. The `tidyverse` includes packages such as `ggplot2`, `dplyr` and `purrr` [Wickham and Grolemund, 2017]. The `sits` makes extensive use of the `tidyverse`.

For a better explanation of how the "`sits` tibble" works, we will read a data set containing 2,115 labelled samples of land cover in Mato Grosso state of Brazil. This state has 903,357 km^2 of extension, being the third largest state of Brazil. It includes three of Brazil's biomes: Amazonia, Cerrado and Pantanal. It is the most important agricultural frontier of Brazil and is Brazil's largest producer of soybeans, corn and cotton.

The samples contain time series extracted from the MODIS MOD13Q1 product from 2000 to 2016, provided every 16 days at 250-meter spatial resolution in the Sinusoidal projection. Based on ground surveys and high resolution imagery, we selected 2,115 samples of nine classes: "Forest", "Cerrado", "Pasture", "Soybean-fallow", "Fallow-Cotton", "Sybean-Cotton", "Soybean-Corn", "Soybean-Millet", and "Soybean-Sunflower".

```
# data set of samples
# print the first three samples
samples_MT_9classes[1:3,]
```

```
## # A tibble: 3 x 7
##   longitude latitude start_date end_date   label   coverage   time_series
##       <dbl>    <dbl> <date>     <date>     <chr>   <chr>      <list>
## 1    -55.2   -10.8  2013-09-14 2014-08-29 Pasture mod13q1_512 <tibble [2~
## 2    -57.8   - 9.76 2006-09-14 2007-08-29 Pasture mod13q1_512 <tibble [2~
```

```
## 3    -51.9   -13.4  2014-09-14 2015-08-29 Pasture mod13q1_512 <tibble [2~
```

The "sits tibble" contains data and metadata. The first six columns contain the metadata: spatial and temporal location, label assigned to the sample, and coverage from where the data has been extracted. The spatial location is given in longitude and latitude coordinates for the "WGS84" ellipsoid. For example, the first sample has been labelled "Pasture", at location (−55.1852, −10.8387), and is considered valid for the period (2013-09-14, 2014-08-29). Informing the dates where the label is valid is crucial for correct classification. In this case, the researchers involved in labelling the samples chose to use the agricultural calendar in Brazil, where the spring crop is planted in the months of September and October, and the autumn crop is planted in the months of February and March. For other applications and other countries, the relevant dates will most likely be different from those used in the example.

The "sits tibble" also contains the time series data for each spatiotemporal location. The timeseries data is also organized as a tibble, with a column with the dates and the other columns with the values for each spectral band.

```
# print the first 10 time series records of the first sample
samples_MT_9classes$time_series[[1]][1:3,]
```

```
## # A tibble: 3 x 7
##   Index       ndvi   evi   nir   mir   blue    red
##   <date>     <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1 2013-09-14 0.424 0.280 0.288 0.244 0.0605 0.116
## 2 2013-09-30 0.467 0.264 0.257 0.167 0.0357 0.0933
## 3 2013-10-16 0.504 0.299 0.266 0.202 0.0405 0.0877
```

The sits package provides functions for data manipulation and displaying information of a "sits tibble". For example, the command sits_labels() that shows the labels of the sample set and their frequencies.

```
sits_labels(samples_MT_9classes)
```

```
## # A tibble: 9 x 3
##   label         count   freq
##   <chr>         <int>  <dbl>
## 1 Cerrado         400 0.189
## 2 Fallow_Cotton    34 0.0161
## 3 Forest          138 0.0652
## 4 Pasture         370 0.175
## 5 Soy_Corn        398 0.188
## 6 Soy_Cotton      399 0.189
## 7 Soy_Fallow       88 0.0416
## 8 Soy_Millet      235 0.111
## 9 Soy_Sunflower    53 0.0251
```

In many cases, it is useful to relabel the data set. For example, there may be situations when one wants to use a smaller set of labels, since samples in one label on the original set may not be distinguishable from samples with other labels. We then should use `sits_relabel()`, which requires a conversion list (for details, see `?sits_relabel`).

Given that we have used the tibble data format for the metadata and and the embedded time series, one can use the functions of the `dplyr`, `tidyr` and `purrr` packages of the `tidyverse` [Wickham and Grolemund, 2017] to process the data. For example, the following code uses the `sits_select()` function to get a subset of the sample data set with two bands (NDVI and EVI) and then uses the `dplyr::filter()` function to select the samples labelled either as "Cerrado" or "Pasture". We can then use the `sits_plot()` function to display the time series. Given a small number of samples to display, the `sits_plot()` function tries to group as many spatial locations together. In the following example, the first 15 samples of the "Cerrado" class all refer to the same spatial location in consecutive time periods. For this reason, these samples are plotted together.

```
# select the "ndvi" bands
samples_ndvi.tb <-
    sits_select(samples_MT_9classes,
                bands = c("ndvi"))
# select only the samples with the cerrado label
samples_cerrado.tb <-
    dplyr::filter(samples_ndvi.tb,
                  label == "Cerrado")
# plot the first 15 samples (different dates for the same points)
sits_plot(samples_cerrado.tb[1:15,])
```

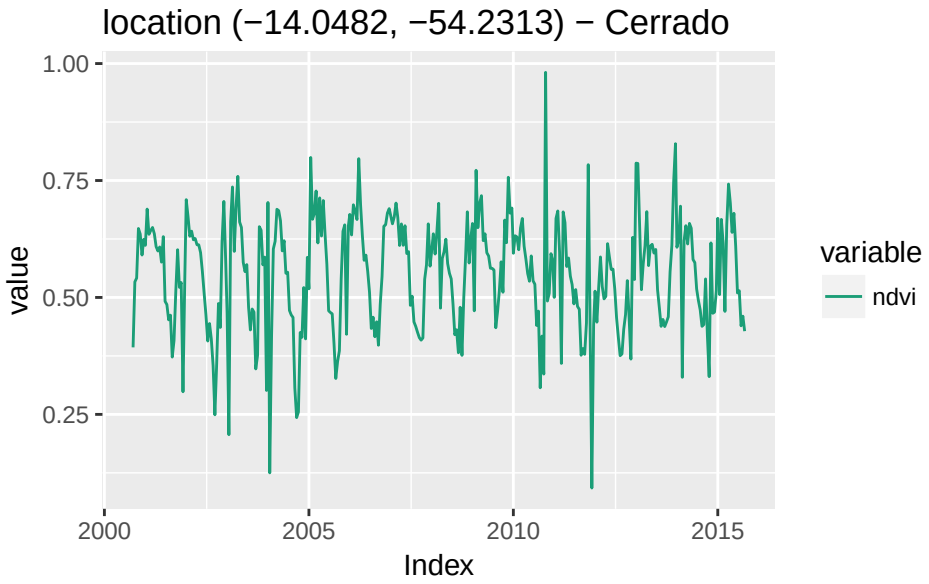


Figure 1: Plot of the first 15 'Cerrado' samples from data set `samples_MT_9classes` (different dates for the same point location).

For a large number of samples, where the amount of individual plots would be substantial, the default visualisation combines all samples together in a single temporal interval (even if they are valid for different years). Therefore, all samples of the same band and the same label are aligned to a common interval. This plot is useful to show the spread of values for the time series of each band. The strong red line in the plot shows the median of the values, and the two orange lines are the first and third interquartile ranges. The `sits_plot()` function has different ways of working. Please, refer to the package documentation for more details.

```
# plot all cerrado samples together (shows the distribution)
sits_plot(samples_cerrado.tb)
```

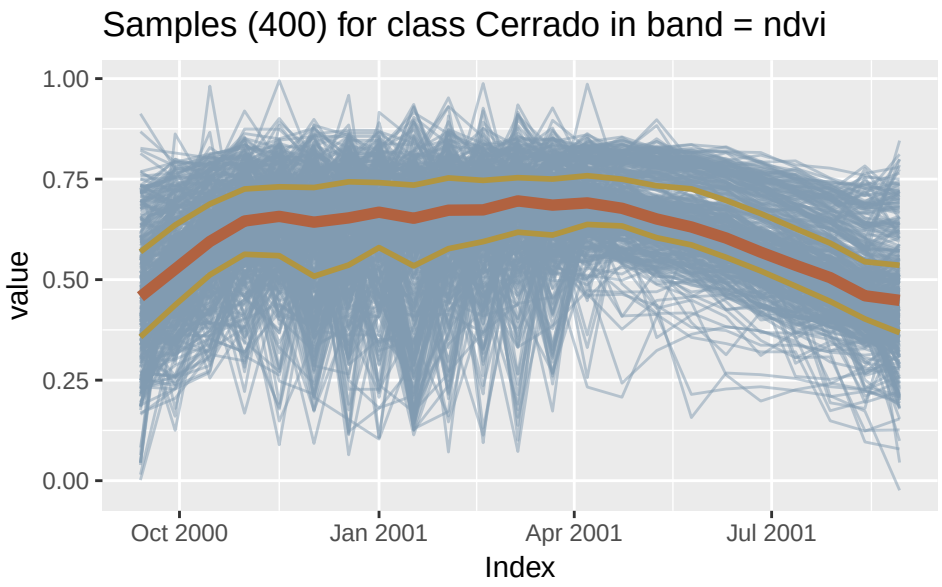


Figure 2: Plot of all 'Cerrado' samples from data set `samples_MT_9classes`.

Usually, samples are provided by experts whose take *in-loco* field observations or recognises land classes through high resolution images. In any case, we need access to a data source to fetch time series data regarding a spatiotemporal location of interest. The process of importing data samples is discussed in the next section.

Importing Data into `sits`

The `sits` package allows different methods of data input, including: (a) obtain data from a WTSS (Web Series Time Service); (b) obtain data from the SATVEG service developed by EMBRAPA (Brazil's Agriculture Research Agency). (c) read data stored in a time series in the ZOO format [Zeileis and Grothendieck, 2005]; (d) read a time series from a RasterBrick [Hijmans, 2015]. Option (d) will be described in the section were we describe raster processing. The WTSS service is a light-weight service, designed to retrieve time series for selected locations and periods [Vinhas et al., 2016], been implemented by the research team of the National Institute for Space Research to allow remote access to

time series data. The SATVEG service provides NDVI and EVI time series vegetation indices from MODIS image from whole Brazilian territory [EMBRAPA, 2014]. To view service details, the user needs to call `sits_services()` that provides information on the coverages available on the server.

After finding out which coverages are available at the different time series services, one may request specific information on each coverage by using `sits_coverage()`. This lists the contents of the data set, including source, bands, spatial extent and resolution, time range, and temporal resolution. This information is then stored in a tibble for later use.

```
# get information about a specific coverage from WTSS
coverage_wtss <-
    sits_coverage(service  = "WTSS-INPE-1",
                  product  = "MOD13Q1",
                  name     = "mod13q1_512")
coverage_wtss[, c("xmin","xmax","ymin","ymax",
                "start_date", "end_date")]
```

```
## # A tibble: 1 x 6
##    xmin  xmax  ymin  ymax start_date end_date
##   <dbl> <dbl> <dbl> <dbl> <date>     <date>
## 1 -81.2 -31.9 -30.0  10.0 2000-02-18 2017-02-18
```

The user can request one or more time series points using `sits_getdata()`. This function provides a general means of access to image time series. In its simplest fashion, the user provides the latitude and longitude of the desired location, the product and coverage names, the bands, and the start date and end date of the time series. If the start and end dates are not provided, all available period is retrived. The result is a tibble that can be visualised using `sits_plot()`.

```
# a point in the transition forest pasture in Northern MT
# obtain a time series from the WTSS server for this point
series.tb <-
    sits_getdata(longitude  = -55.57320,
                 latitude   = -11.50566,
                 coverage   = coverage_wtss,
                 bands      = c("ndvi", "evi"))
# plot the series
sits_plot(series.tb)
```

A useful case is when users have a set of labelled samples, that are to be used as a training data set. In this case, one usually has trusted observations which are labelled and commonly stored in plain text CSV files. The `sits_getdata()` function can receive a CSV file path as an argument. The CSV file must provide for each time series, its latitude and longitude, the start and end dates, and a label associated to a ground sample.

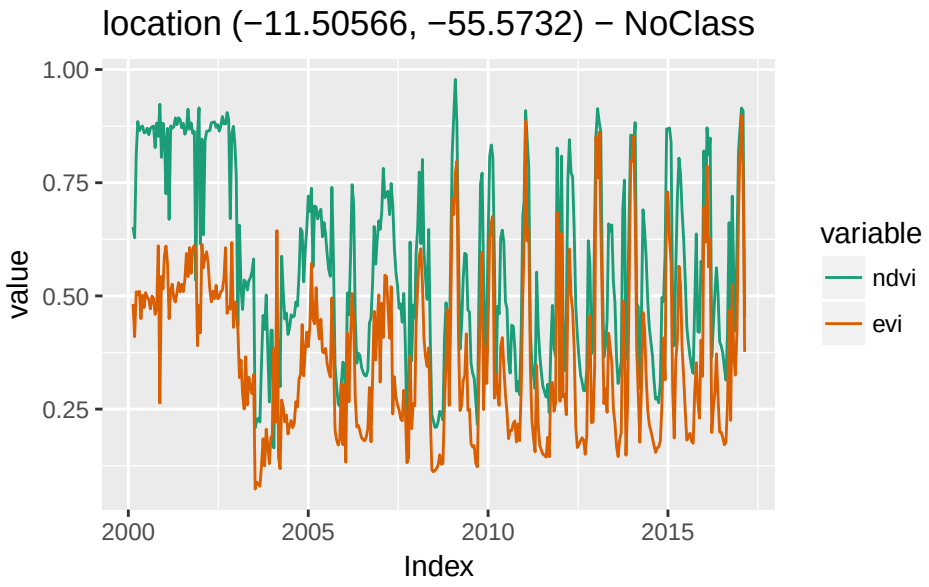


Figure 3: NDVI and EVI time series fetched from WTSS service.

After importing the samples time series, it is useful to explore the data and see how is it underlying structured and its inter-class separability. For example, We can note in the figure above the variability of 400 time series samples along time. Those samples were collected from different years and/or locations. The scattering behaviour is intrinsic to remote sensing data. Atmospheric noise, sun angle, interferences on observations or different equipaments specifications, as well as the very nature of the climate-land dynamics can be sources of such variability [Atkinson et al., 2012]. One helpful technique to explore such properties is *cluster analysis*. In the following section we present a cluster technique supported by `sits`.

Clustering in satellite image time series

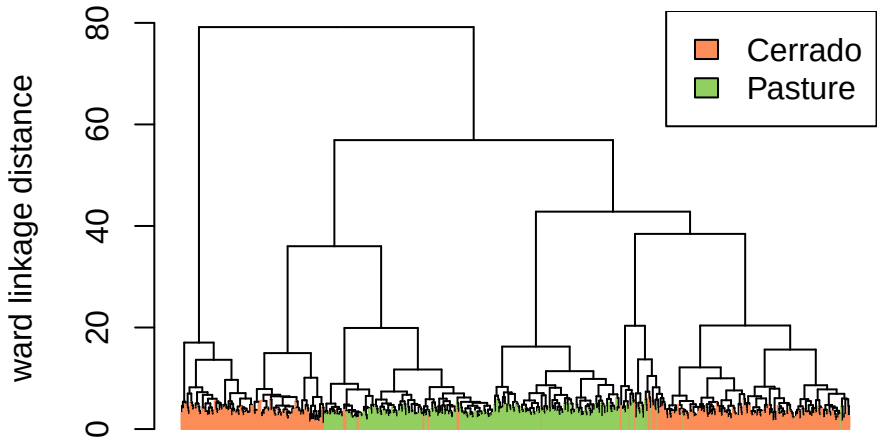
Cluster analysis has been used for many purposes in satellite image time series literature ranging from unsupervised classification [Petitjean et al., 2011], and pattern detection [Romani et al., 2011]. Here, we are interested in the second use of clustering, as a way to improve training data to use in machine learning classification models. In this regard, cluster analysis can assist the identification of structural *time series patterns*, and anomalous samples [Romani et al., 2011], [Chandola et al., 2009]. `sits` provides support for the agglomerative hierarchical clustering (AHC).

Hierarchical clustering is a family of methods that groups elements using a distance function to associate a real value to a pair of elements. From this distance measure, we can compute the dissimilarity between any two elements from the data set. Depending on the distance functions and linkage criteria, the algorithm decides which two clusters are merged at each iteration. The AHC approach is suitable for the purposes of samples data exploration awe its visualisation power and ease of use [Keogh et al., 2003]. Moreover, AHC does not require a predefined number of clusters as an initial parameter. This is an

important feature in satellite image time series clustering since it is not easy to define the number of clusters present in a set of multi-attribute time series [Aghabozorgi et al., 2015].

The main result of AHC method is the *dendrogram*. A *dendrogram* is the ultrametric relation formed by the successive merges in the hierarchical process that can be represented by a tree. Dendrograms are quite useful to decide on the number of clusters has the data. It shows the height where each merging happened, which corresponds to the minimum distance between two clusters defined by a *linkage criterion*. The most commom linkage criteria are: *single-linkage*, *complete-linkage*, *average-linkage*, and *Ward-linkage*. Complete-linkage prioritises the within-cluster dissimilarities, producing clusters with shorter distance samples. Complete-linkage clustering can be sensitive to outliers, that can increase the resulting intracluster data variance. As an alternative, Ward proposes a criteria to minimise the data variance by means of either *sum-of-squares* or *sum-of-squares-error* [Ward, 1963]. Ward's intuition is that clusters of multivariate observations, such as time series, should be approximately elliptical in shape [Hennig, 2015]. In `sits`, a dendrogram can be generated by `sits_dendrogram()`. The following codes illustrate how to create, visualise, and cut a dendrogram (for details, see `?sits_dendrogram()`).

```
# take a set of patterns for 2 classes
# create a dendrogram object with default clustering parameters
dendro <- sits_dendrogram(cerrado_2classes)
# plot the resulting dendrogram
sits_plot_dendrogram(cerrado_2classes,
                     dendro)
```



After the creation of a dendrogram, an important question emerges: *where to cut the dendrogram?* The answer depends on what are the purposes of the cluster analysis [Hennig, 2015]. If one is interested in an unsupervised classification, it is commom to use *internal validity indices*, such as Silhouettes [Rousseeuw, 1987], to help determine the best number of clusters. However, if one is interested in understand the structure of a labeled data set, or in the identification of sample anomaly, as we are here, one can reccur to *external validity indices* to assist the semisupervised procedure that achieves

the optimal correspondence between the clusters and classes partitions. In this regard, we need to balance two objectives: get clusters as large as possible, and get clusters as homogeneous as possible with respect to its known classes. To help this process, `sits` provides `sits_dendro_bestcut()` function that computes the external validity index *adjusted Rand index* (ARI) for a series of different number of generated clusters. The function returns the height where the cut of the dendrogram maximizes the index.

```
# search for the best height to cut the dendrogram
sits_dendro_bestcut(cerrado_2classes,
                    dendro)
```

```
##        k   height
##  6.00000 20.39655
```

This height optmises the ARI and generates 6 clusters. The ARI considers any pair of distinct samples and computes the following counts: a) the number of distinct pairs whose samples have the same label and are in the same cluster; b) the number of distinct pairs whose samples have the same label and are in different clusters; c) the number of distinct pairs whose samples have different labels and are in the same cluster; d) the number of distinct pairs whose samples have the different labels and are in different clusters. Here, a and d consists in all agreements, and b and c all disagreements. The ARI is obtained by

$$ARI = \frac{a+d-E}{a+d+b+c-E},$$

where E is the expected agreement, a random chance correction calculated by

$$E = (a+b)(b+c)+(c+d)(b+d).$$

Different from others validity index such as Jaccard ($J = a/(a+b+c)$), Fowlkes-Mallows ($FM = a/(a^2+a(b+c)+bc)^{1/2}$), and Rand (the same as ARI without the E adjustment) indices, ARI is more appropriate either when the number of clusters is outwheighted by the number of labels (and *vice versa*) and the amount of samples in labels and clusters is imbalanced [Hubert and Arabie, 1985], which is usually the case.

```
# create 6 clusters by cutting the dendrogram at
# the linkage distance 20.39655
clusters.tb <-
    sits_cluster(cerrado_2classes,
                 dendro,
                 k = 6)
# show clusters samples frequency
sits_cluster_frequency(clusters.tb)
```

```
##
##              1   2   3   4   5   6 Total
##   Cerrado  203  13  23  80   1  80   400
##   Pasture    2 176  28   0 140   0   346
##   Total    205 189  51  80 141  80   746
```

Note in this example that almost all clusters has a predomination of either "Cerrado" or "Pasture" classes with the exception of cluster 3. The contigency table ploted by `sits_cluster_frequency` shows how the samples are distributed across the clusters and helps to identify two kinds of confusions. The first is relative to those small amount of samples in clusters dominated by another class (*e.g.* clusters 1, 2, 4, 5, and 6), and the second is relative to those samples in non-dominated clusters (*e.g.* cluster 3). These confusions can be an indication of poor quality of samples, or an inadequacy between the used parameters in cluster analysis, or even a natural confusion due to the inherent variability of the land classes.

For whatever reason, one can check other methods to assist one's decision, either by eliminating part of it, or by improving the data set. If one considers such cases as outliers, it is possible to remove them using the functions `sits_cluster_clean()` and `sits_cluster_remove()`. The first removes all those minority samples that do not reach a minimum percentage close to 0%, whereas the second removes an entire cluster if its dominant class does not reach a minimum percentage, close to 100%. The example illustrates the second aproach.

```
# clear those samples with a high confusion rate in a cluster
# (those clusters which majority class does not reach 90% of
#  samples in that cluster)
cleaned.tb <-
    sits_cluster_remove(clusters.tb,
                        min_perc = 0.9)
# show clusters samples frequency
sits_cluster_frequency(cleaned.tb)
```

```
##
##              1   2   4   5   6 Total
##   Cerrado  203  13  80   1  80   377
##   Pasture    2 176   0 140   0   318
##   Total    205 189  80 141  80   695
```

Along the process of cluster analysis, it may be a good practice to measure the correspondence between clusters and labels partitions through computation of external validity indices. These measures can help the comparison among different procedures and assits the decision-making. `sits_cluster_validity()` provides a way to compute some external validation indices other than ARI (for details, see `?sits_cluster_validity()`). Moreover, these indices capture some of the cluster structure that is present in the correspondence of its partitions [Hubert and Arabie, 1985].

Filtering techniques

Satellite image time series generally is contaminated by atmospheric influence, geolocation error, and directional effects [Lambin and Linderman, 2006]. Inter-annual climate variability also changes the phenological cycles of the vegetation, resulting in time series

whose periods and intensities do not match on an year to year basis. To make the best use of available satellite data archives, methods for satellite image time series analysis need to deal with data sets that are *noisy* and *non-homogeneous*. In this section we discuss filtering techniques, a method used to improve time series data that presents missing values or noise.

The literature on satellite image time series have several applications of filtering used to correct or smooth vegetation index data. The `sits` have support for Savitzky–Golay (`sits_sgolay()`), Whitaker (`sits_whittaker()`), envelope (`sits_envelope()`) and, the "cloud filter" (`sits_cloud_filter()`) filters. The first two filters are commonly used in the literature. The remaining two are adaptations of other models and, for our knowledge, its use have not been reported in the literature.

Various somewhat conflicting results have been expressed in relation to the time series filtering techniques for phenology applications. For example, in an investigation of phenological parameter estimation, Atkinson et al. [2012] found that the Whittaker and Fourier transform approaches were preferable to the double logistic and asymmetric Gaussian models. They applied the filters to preprocess MERIS NDVI time series for estimating phenological parameters in India. Comparing the same filters as in the previous work, Shao et al. [2016] found that only Fourier transform and Whittaker techniques improved interclass separability for crop classes and significantly improved overall classification accuracy. The authors used MODIS NDVI time series from the Great Lakes region in North America. Zhou et al. [2016] found that asymmetric Gaussian model outperforms other filters over high latitude boreal biomes, while the Savitzky-Golay model gives the best reconstruction performance in tropical evergreen broadleaf forests. In the remaining biommes, Whittaker gives superior results. The authors compare all previous mentioned filters plus Savitzky-Golay method for noise removal in MODIS NDVI data from sites spreaded worldwide in different climatological conditions. Many other techniques can be found in applications on satelite image time series such as curve fitting [Bradley et al., 2007], wavelet decomposition [Sakamoto et al., 2005], and mean-value iteration, ARMD3-ARMA5, and 4253H [Hird and McDermid, 2009] techniques. This shows that comparative analysis of smoothing algorithms depends on the performance measure adopted by the authors.

One of the main uses of time series filtering is to reduce the noise and miss data produced by clouds in tropical areas. The following examples use data produced by the PRODES project [INPE, 2017], which detects deforestation in the Brazilian Amazon rain forest by visual interpretation. `sits` provides 617 samples from a region corresponding to the standard Landsat Path/Row 226/064. This is an area in the East of the Brazilian Pará state and has been chosen because its strongly cloud cover from November to March, which is a significant factor in degrading time series quality. Its NDVI and EVI time series were extracted from a combination of MOD13Q1 and Landsat8 images (to best visualize the effects of each filter, we selected only NDVI time series).

Savitzky–Golay filter

The Savitzky-Golay filter works by fitting a successive array of $2n+1$ adjacent data points with a d-degree polynomial through linear least squares. The central point i of the window array assumes the value of the interpolated polynomial. An equivalent and much faster solution than this convolution procedure is given by the closed expression

$$\hat{x}_i = \sum_{j=-n}^{n} C_j \, x_{i+j},$$

where $\hat{x}$ is the the filtered time series, C_j are the Savitzky-Golay smoothing coefficients, and x is the original time series.

The coeficients C_j depend uniquely on the polynomial degree (d) and the length of the window data points (given by parameter n). If $d = 0$, the coeficients are constants $C_j = 1/(2n+1)$ and the Savitzky-Golay filter will be equivalent to moving average filter. When the time series is equally spaced, the coefficients have analytical solution. According to Madden [1978], for $d \in [2,3]$ each C_j smothing coefficients can be obtained by

$$C_j = \frac{3(3n^2+3n-1-5j^2)}{(2n+3)(2n+1)(2n-1)}.$$

In general, the Savitzky-Golay filter produces smoother results for a larger value of n and/or a smaller value of d [Chen et al., 2004]. The optimal value for these two parameters can vary from case to case. For example, Zhou et al. [2016] set $d = 2$ and $n = 10$. Hird and McDermid [2009] tests the filter for $n \in [5,6,7]$ using quadratic polynomial.

`sits` Savitzky-Golay function The following example shows the effect of Savitsky-Golay filter on the original time series.

```
# Take the NDVI band of the first sample data set
point.tb <-
    sits_select(prodes_226_064[1,],
                bands = c("ndvi"))
# apply SavitzkyGolay filter
point_sg.tb <- sits_sgolay(point.tb)
# plot the series
sits_plot(sits_merge(point_sg.tb, point.tb))
```

Whittaker filter

The Whittaker smoother attempts to fit a curve that represents the raw data, but is penalized if subsequent points vary too much [Atzberger and Eilers, 2011]. The Whittaker filter is a balancing between the residual to the original data and the "smoothness" of the fitted curve. The residual, as measured by the sum of squares of all n time series points deviations, is given by

$$RSS = \sum_i (x_i - \hat{x}_i)^2,$$

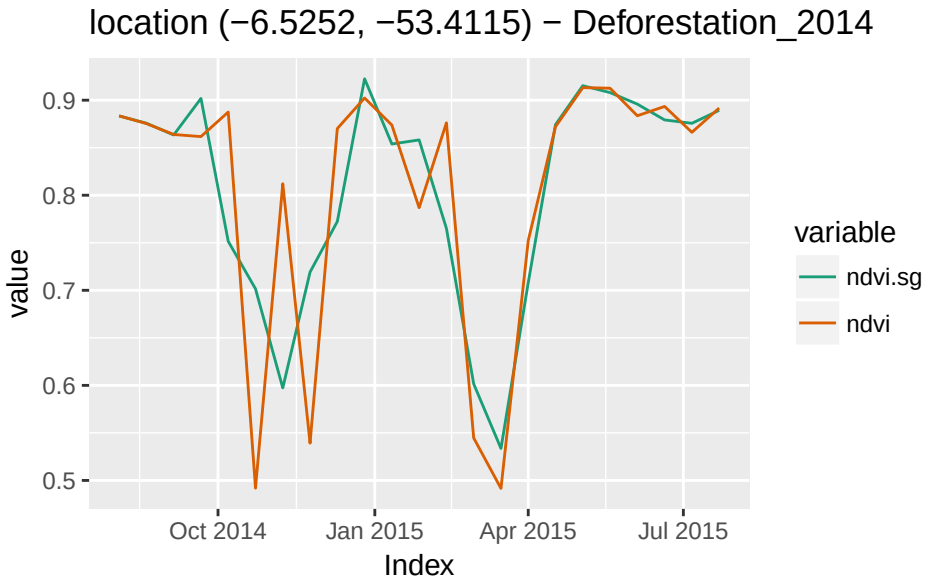


Figure 4: Savitzky-Golay filter applied on a one-year NDVI time series.

where x and $\hat{x}$ are the original and the filtered time series vectors, respectivelly. The smoothness is assumed to be the measure of the the sum of the squares of the third order differences of the time series [Whittaker, 1922] which is given by

$$SSD_3 = (\hat{x}_4 - 3\hat{x}_3 + 3\hat{x}_2 - \hat{x}_1)^2 + (\hat{x}_5 - 3\hat{x}_4 + 3\hat{x}_3 - \hat{x}_2)^2 + \ldots + (\hat{x}_n - 3\hat{x}_{n-1} + 3\hat{x}_{n-2} - \hat{x}_{n-3})^2.$$

Whittaker filter is obtained by finding a new time series $\hat{x}$ whose points minimize the expression

$$RSS + \lambda SSD_3,$$

where λ, a scalar, works as an "smoothing wheight" parameter. The minimization can be obtained by differentiating the expression with respect to $\hat{x}$ and equating it to zero. The solution of the resulting linear system of equations gives the filtered time series which, in matrix form, can be expressed as

$$\hat{x} = (\mathrm{I} + \lambda D^{\intercal} D)^{-1} x,$$

where I is the identity matrix and

$$D = \begin{bmatrix} 1 & -3 & 3 & -1 & 0 & 0 & \cdots \\ 0 & 1 & -3 & 3 & -1 & 0 & \cdots \\ 0 & 0 & 1 & -3 & 3 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

is the third order difference matrix. The Whitakker filter can be a large but sparse optimisation problem, as we can note from D matrix.

Whittaker smoother has been used only recently in satellite image time series investigations. According to Atzberger and Eilers [2011], the smoother has an advantage over other filtering techniques such as Fourier and wavelets as it does not assume signal

periodicity. Moreover, the authors argues that Whittaker permits a rapid processing of large amounts of data, and handles easily incomplete time series with missing values. The fact that it has only one parameter (λ) facilitates its calibration/comparison process. Zhou et al. [2016] found that $\lambda = 15$ gives the best result when compared with $\lambda = 2$. Larger values of λ produces smoother results.

```
# Take the NDVI band of the first sample data set
point.tb <-
    sits_select(prodes_226_064[1,],
                bands = c("evi"))
# apply Whitaker filter
point_whit.tb <- sits_whittaker(point.tb)
# plot the series
sits_plot(sits_merge(point_whit.tb,
                     point.tb))
```

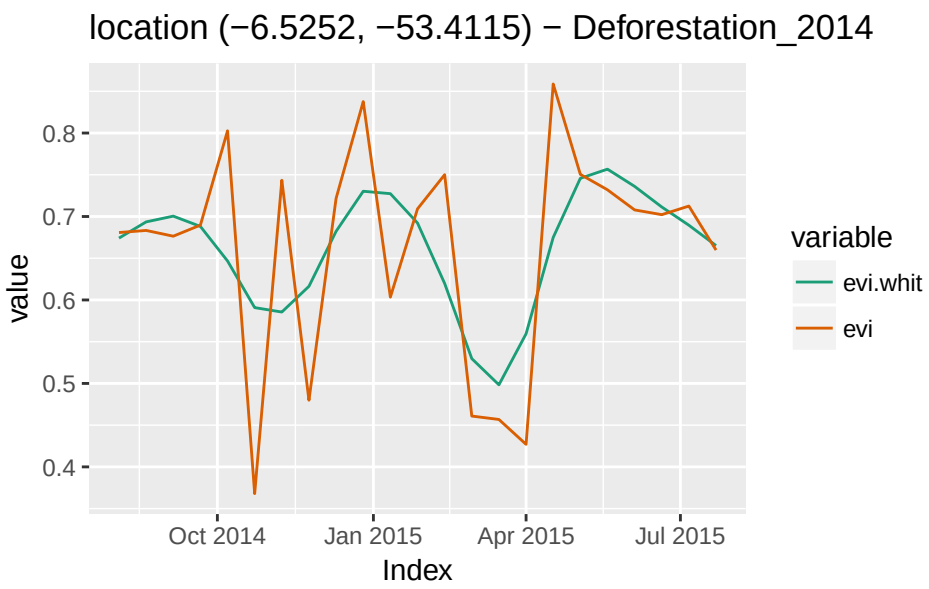


Figure 5: Whittaker smoother filter applied on one-year NDVI time series. The example uses default $\lambda = 1$ parameter.

Envelope filter

This filter can generate a time series corresponding to the superior (inferior) bounding of the input signal. This is acomplished through a convoluting window (odd length) that attributes to the point i, in the resulting time series, the maximimum (minimum) value of the points in the window. The i point corresponds to the central point of the window. It can be defined as

$$u_i = \max_k(\{x_k : |i-k| \leq 1\}),$$

whereas an lower dilation is obtained by

$$l_i = \min_k(\{x_k : |i-k| \leq 1\}).$$

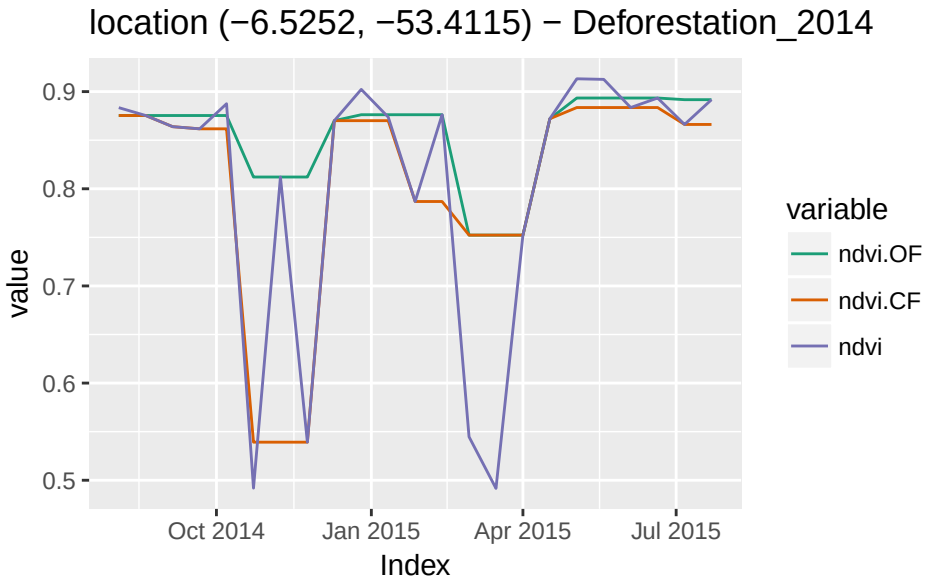
Here, x is the input time series and, k and i are vector indices.

The `sits_envelope()` can combine both maximum and minimum window sequencially. The function can receive a string sequence with `"U"` (for maximization) and `"L"` (for minimization) characters passed to its parameter. A repeated sequence of the same character is equivalent to one operation with a larger window. The sequencial operations on the input time series produces the final filtered result that is returned.

The envelope filter can be viewed through mathematical morphology lenses, a very common field in digital image processing [Haralick et al., 1987]. Here the operations of `"U"` and `"L"` corresponds to the *dilation* and *erosion* morphological operators applied to univariate arrays [Vávra et al., 2004]. Furthermore, the compounds operation of *opening* and *closing* can be obtained by `"UL"` and `"LU"`, respectivelly. This technique has been applied on time series analysis in other fields [Accardo et al., 1997] but, for our knowledge, there is no application in satellite image time series literature.

In the following example we can see an application of `sits_envelope()` function. There, we performs the *opening filtration* and *closing filtration* introduced by Vávra et al. [2004]. The correspondent operations sequence are `"ULLULUUL"` and `"LUULULLU"`.

```
# Take the NDVI band of the first sample data set
point.tb <-
    sits_select(prodes_226_064[1,],
                bands = c("ndvi"))
# apply envelope filter (remove downward and upward noises)
point_env1.tb <-
    sits_envelope(point.tb,
                  "ULLULUUL",
                  bands_suffix = "OF")
point_env2.tb <-
    sits_envelope(point.tb,
                  "LUULULLU",
                  bands_suffix = "CF")
# plot the series
sits_plot(
    sits_merge(
        sits_merge(point_env1.tb,
                   point_env2.tb),
        point.tb))
```

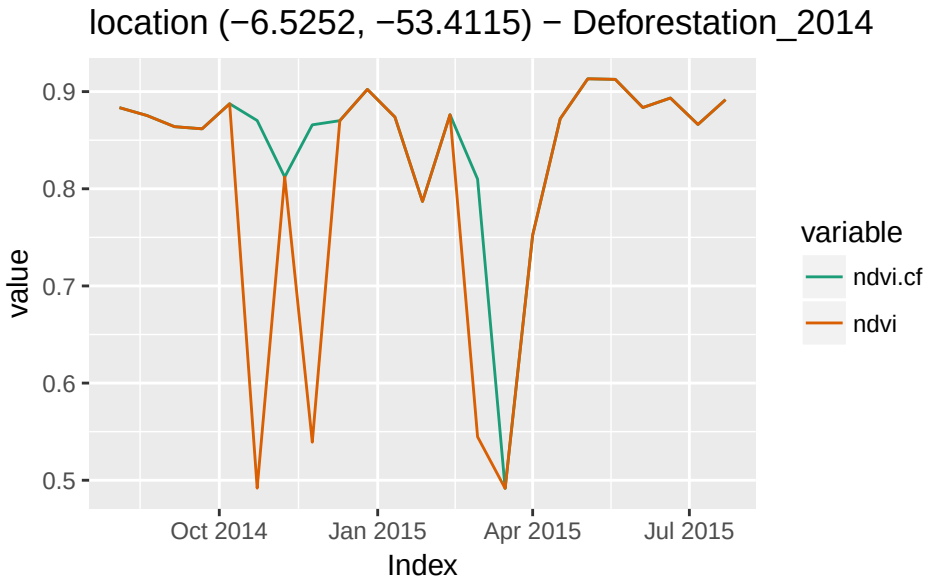


Cloud filter

The cloud filter makes use of the well known autoregressive integrated moving average (ARIMA) model. The algorithm looks to the first order difference time series for points where the value is above a certain threshold. This procedure selects only those points with large variations in the original time series, probably associated with noise. Finally, these points are replaced by the ARIMA correspondent values.

The parameters of the ARIMA model can be set by the user. Please see arima for the detailed description of parameters p, d, and q.

```
# Take the NDVI band of the first sample data set
point.tb <- sits_select(prodes_226_064[1,], bands = c("ndvi"))
# apply ARIMA filter
point_cf.tb <- sits_cloud_filter(point.tb, apply_whit = FALSE)
# plot the series
sits_plot(sits_merge(point_cf.tb, point.tb))
```



Machine learning classification for land use and land cover using satelite image time series

The main advantage using satellite image time series in land use studies is that the time series is methodologically consistent with the very nature of the land covers. Using this kind of data allows focusing on land changes through time. Currently, most studies that use satellite image time series for land classification still use variations of the classical remote sensing image classification methods. Given a series of images, researchers use methods that produce a single composite for the whole series [Gomez et al., 2016]. In their review on this subject, Gomez et al. [2016] discuss 12 papers that use satellite image time series to derive image composites that are later used for classification. Câmara et al. [2016] denote these works as taking a *space-first, time-later* approach.

An example of *space-first, time-later* work on big EO data analysis is the work by Hansen et al. [2013]. Using more than 650,000 Landsat images and processing more than 140 billion pixels, the authors compared data from 2000 to 2010 to produce maps of global forest loss during the decade. A pixel-based classification algorithm was used to process each image to detect forest cover. The method classifies each 2D image one by one.

In our view, these methods do not use the full potential of satellite image time series. The benefits of remote sensing time series analysis arise when the temporal resolution of the big data set is able to capture the most important changes. Here, the temporal autocorrelation of the data can be stronger than the spatial autocorrelation. Given data with adequate repeatability, a pixel will be more related to its temporal neighbours than to its spatial ones. In this case, *time-first, space-later* methods lead to better results than the *space-first, time-later* approach [Câmara et al., 2016].

The `sits` package provides functionality to explore the full depth of satellite image time series data. It treat time series as a feature vector. To be consistent, the procedure aligns all time series from different years by its time proximity considering an given

cropping schedule. Once aligned, the feature vector is formed by all pixel "bands". The idea is to have as many temporal attributes as possible, increasing the dimension of the classification space. In this scenario, statistical learning models are the natural candidates to deal with high-dimensional data: learning to distinguish all land cover and land use classes from trusted samples exemplars (the training data) to infer classes of a larger data set. `sits` has support for a variety of machine learning techniques (...). In the following sections we discuss on SVM and Random Forest machine learning techniques with more detail.

Support Vector Machine

The Support Vector Machine (SVM) is a generalization of the *separating hyperplane classifier* [Hastie et al., 2009]. This generalization combines the notions of the *optimal separating hyperplanes*, the *soft margins*, and the *enlargement of the input attribute space* with a nonlinear mapping to a feature space.

In a separating hyperplane formulation, the classifier works only with separable sample sets. It finds a linear boundary in the input attribute space that not only divides two classes but maximizes its margin to them. A boundary is a *hyperplane* that divides the entire attribute space into two parts. Let $(\mathbf{x}_k, y_k)$, $k \in (1, \ldots, n)$, be a set of n observations with attribute vector $\mathbf{x}_k \in \mathbb{R}^p$ and class $y \in \{-1, 1\}$. Once feature space has p dimensions, the hyperplane has dimension $p-1$ and can be written as

$$a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_p x_p = 0,$$

where x_i is the i-th attribute of $\mathbf{x}$.

If the samples are separable, the hyperplane has a perfect classification property, that is, for all sample $(\mathbf{x}_k, y_k)$, the following relation holds

$$\begin{aligned} a_0 + a_1 x_{k_1} + a_2 x_{k_2} + \ldots + a_p x_{k_p} < 0, \quad y_k = -1, \\ a_0 + a_1 x_{k_1} + a_2 x_{k_2} + \ldots + a_p x_{k_p} > 0, \quad y_k = 1. \end{aligned}$$

Once the solution for a such hyperplane is obtained (*i.e.* values for coeficients $a_0, a_1, \ldots, a_p$), one is able to derive a very simple classification model

$$f(\mathbf{x}) = \hat{a}_0 + \hat{\mathbf{a}}\mathbf{x}$$

that classifies any test vector $\mathbf{x}$ according to

$$\begin{aligned} y = 1, \quad \text{if } f(\mathbf{x}) > 0, \\ y = -1, \quad \text{if } f(\mathbf{x}) < 0. \end{aligned}$$

However, the condition under which this classifier has solution is too restrictive, as it needs a linearly separable training set. This is hardly what one may expect in a typical satellite image time series scenario. To overcome this undesired constraint, SVM introduces a *softner term* in the hyperplane optimization problem. This modification allows some observations to be inside the margins band, or even in the incorrect side of the boundary, violating the maximum margin criteria but increasing its robustness by

reducing individual observations influence. Moreover, the solution for the hyperplane coefficients $\hat{\mathbf{a}} = (a_1, \ldots, a_n)^\intercal$, given by

$$\hat{\mathbf{a}} = \sum_{k=1}^{n} \alpha_k y_k \mathbf{x}_k,$$

depends only on those samples that violates the maximum margin criteria, here captured by the fact that $\alpha_k > 0$ only for those points laying either inside the margin band or in the wrong side of the boundary, the so-called *support vectors*. For all other points, $\alpha_k = 0$, and hence none of them exert any influence on the coefficients determination. This property let SVM less sensitive to outliers (for more details on how to compute α and $\hat{a}_0$, see Cortes and Vapnik [1995], Hastie et al. [2009], and James et al. [2013]).

Hyperplanes are linear $(p-1)$-dimensional boundaries and define linear partitions in the attribute space. However, sometimes we are faced with nonlinear relationship between attribute vectors and classes. In such cases, linear classification models can suffer to do a good job. An alternative to this limitation is the *enlargement of the attribute space* by a nonlinear mapping ϕ to a higher degree feature space. In this manner, the new classification model

$$f(x) = \hat{a}_0 + \hat{\mathbf{a}}^\intercal \phi(\mathbf{x}),$$

defines its linear hyperplane on the enlarged feature space which translate to nonlinear boundaries in the original input space. Here, the coefficients are obtained simmilarly by

$$\hat{\mathbf{a}} = \sum_{k=1}^{n} \alpha_k y_k \phi(\mathbf{x}_k),$$

which substituting in f expression above gives us

$$f(x) = \hat{a}_0 + \sum_{k=1}^{n} \alpha_k y_k \phi(\mathbf{x}_k)^\intercal \phi(\mathbf{x}).$$

Moreover, the linearity of the model implies that the term $\phi(\mathbf{x}_k)^\intercal \phi(\mathbf{x})$ can be expressed as a dot product $\phi(\mathbf{x}_k) \cdot \phi(\mathbf{x})$ without change its properties.

However, the enlargement of the feature space by ϕ can be computationaly expensive. SVM makes use of *kernels* to overcome this limitation. Kernel functions are equivalent to dot products of a larger degree feature space vectors. Hence, we can express f as [Cortes and Vapnik, 1995]:

$$f(\mathbf{x}) = \hat{a}_0 + \sum_{k=1}^{n} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}),$$

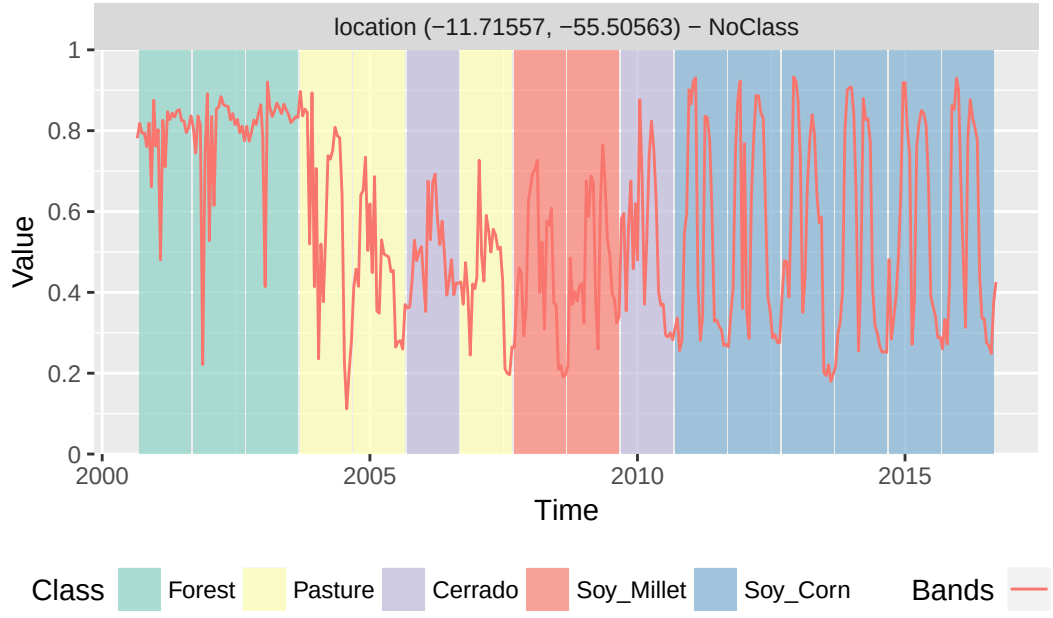
where $K(u, v)$ is a kernel function that is equivalent to the dot product $\phi(u) \cdot \phi(v)$ (see Cortes and Vapnik [1995]). It follows that, we can use kernel functions without the need to compute ϕ, otherwise it would require massive computations. For example, the radial basis function (RBF) kernel

$$K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma \|\mathbf{u} - \mathbf{v}\|^2),$$

one of the most used nonlinear kernels, maps the input space to an infinite dimensional feature space, which would be roughly obtained by a power series expansion with substantially more calculations. Other popular choices for K in the SVM literature are the polynomial kernel ($K(\mathbf{u},\mathbf{v}) = (\gamma \mathbf{u}^\intercal \mathbf{v} + r)^d$, with parameters γ, r, and d), and the linear kernel $K(\mathbf{u},\mathbf{v}) = \mathbf{u}^\intercal \mathbf{v}$. The use of kernels are an efficient computational strategy to produce nonlinear boundaries in the input attribute space generally achieving better training-class separation.

In `sits`, SVM is the default machine learning model. As a wrapper of `e1071` R package that uses the `LIBSVM` implementation [Chang and Lin, 2011], `sits` adopts the *one-against-one* method for multiclass classification. For a q class problem, this method creates $q(q-1)/2$ SVM binary models, one for each class pair combination and tests any unknown input vectors throughout all those models. The overeall result is computed by a voting scheme. The following example ilustrate how to classify an individual sample (in this case a serie of 16 one-year NDVI time series from the same location). We used the NDVI time series from Mato Grosso Brazilian state as a training data set.

```
# Retrieve the set of samples (provided by EMBRAPA) from the
# Mato Grosso region for train the SVM model
data(samples_MT_ndvi)
# get a point to be classified
data(point_ndvi)
# Classify using SVM model
class.tb <-
    sits_classify(point_ndvi,
                  samples_MT_ndvi,
                  ml_method = sits_svm(kernel = "radial",
                                       cost = 10))
sits_plot(class.tb)
```



Random forest

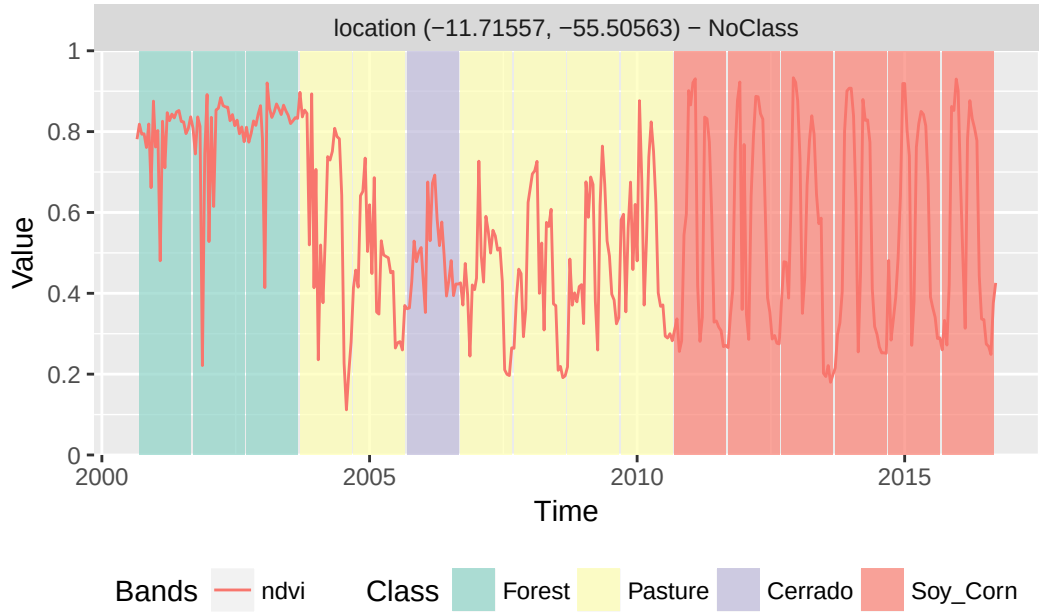
Random forests can be used either as a regression or as a classification technique. Here, we are interested in the latter case. Random forests combines two main ideas that emerged in 1990s statistical learning literature: developing population of trees from *stochastic feature selection* [Ho, 1995], and *bagging* [Breiman, 1996]. It is an *ensemble learning* technique that develops a population of somewhat uncorrelated decision trees that are combined to form a predicting model which is made by a majority vote schema. This aggregating procedure substantialy improves the predictive power of individual trees. Unlike SVM, random forest undertakes a nondeterministic model fitting.

It starts by creating trees through a bootstrap procedure, that is, a repeatedly random samplings with replacement from which, at every iteration, a decision tree is produced considering only that fraction of the data. Each growing tree splits its nodes by a *stochastic feature selection*. At each node splitting, it selects m attributes at random from the existing full set of p attributes ($m < p$). The best attribute is selected according to a given criteria among the m candidates, which then is used to split the nodes at some level. Generally, one can use any purity criterion such as Gini or entropy to decide which attribute is to be used to split the node.

The stochastic feature selection helps to decorrelate the decision trees produced in the bootstrap. This overall procedure generates a set of b somewhat decorrelated decision trees which forms the final random forest model. The prediction is then made by simple consensus: applying each tree to a given test observation vector and computing the predicted class as one vote we choose the most commonly occurring class among the b predictions as the random forest model prediction.

Random forest provides better performances over *bagged trees*, a similar procedure that does not implement stochastic feature selection (*i.e.* when $m = p$). Bagged trees suffer from high correlation among trees introduced by an eventual presence of strong predictors that are deterministically chosen as splitting nodes criterion [James et al., 2013]. However, the random forest classification performance can vary according to the tunning of the model parameters. The main parameters are the number of attributes randomly sampled as candidates at each split (m), the number of decision trees to grow (b), the minimum size of final nodes (), and the sample size to draw at each bootstrap iteration ().

```
# Retrieve the set of samples (provided by EMBRAPA) from the
# Mato Grosso region for train the Random Forest model.
data(samples_MT_ndvi)
# get a point to be classified
data(point_ndvi)
# Classify using Random Forest model
class.tb <-
    sits_classify(point_ndvi,
                  samples_MT_ndvi,
                  ml_method = sits_rfor())
sits_plot(class.tb)
```



Validation techniques

Validation is a process undertaken on models to estimate some error associated with them, and hence has been used widely in different scientific disciplines. Here, we are interested in estimating the prediction error associated to some model. For this purpose, we concentrate on the *cross-validation* approach, probably the most used validation technique [Hastie et al., 2009].

To be sure, cross-validation estimates the expected prediction error. It uses part of the available samples to fit the classification model, and a different part to test it. The so-called *k-fold* validation, we split the data into k partitions with approximately the same size and proceed by fitting the model and testing it k times. At each step, we take one distinct partition for test and the remaining $k-1$ for training the model, and calculate its prediction error for classifing the test partition. A simple average gives us an estimation of the expected prediction error.

A natural question that arises is: *how good is this estimation?* According to Hastie et al. [2009], there is a bias-variance trade-off in choice of k. If k is set to the number of samples, we obtain the so-called *leave-one-out* validation, the estimator gives a low bias for the true expected error, but produces a high variance expectation. This can be computational expensive as it requires the same number of fitting process as the number of samples. On the other hand, if we choose $k = 2$, we get a high biased expected prediction error estimation that overestimates the true prediction error, but has a low variance. The recommended choices of k are 5 or 10 [Hastie et al., 2009], which somewhat overestimates the true prediction error.

`sits_kfold_validate` gives support the k-fold validation in `sits`. The following code gives an example on how to proceed a k-fold cross-validation in the package. It perform a five-fold validation using SVM classification model as a default classifier.

```
# read a set of samples
data (cerrado_2classes)

# perform a five fold validation with the
# SVM machine learning method using default parameters
prediction.mx <-
    sits_kfold_validate(cerrado_2classes,
                        folds = 5)
# prints the output confusion matrix and statistics
sits_conf_matrix(prediction.mx)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Cerrado Pasture
##    Cerrado     398       8
##    Pasture       2     338
##
##           Accuracy : 0.9866
##             95% CI : (0.9755, 0.9936)
##
##              Kappa : 0.973
##
##  Prod Acc  Cerrado : 0.9950
##  Prod Acc  Pasture : 0.9769
##  User Acc  Cerrado : 0.9803
##  User Acc  Pasture : 0.9941
##
```

Raster classification

The continuous observation of the Earth surface provided by orbital sensors is unprecedented in history. Just for the sake of ilustration, a unique tile from MOD13Q1 product, a square of 4800 pixels provided every 16 days since February 2000 takes around 18GB of uncompressed data to store only one band or vegetation index. This data deluge puts the field into a big data era and imposes challenges to design and build technologies that allow the Earth observation community to analyse those data sets [Câmara et al., 2017]. `sits` implements an "out of the box" classification scheme based on *raster cubes* where stacked images have spatial and time dimensions.

Our tested approach and bellow illustrated example is `GeoTIFF` cubes, where the temporal dimension is stored in the image bands. The classification task can be realized in pararel, where the data is consumed piecewise. This strategy enables `sits` to work on desktop computers without depleting all computational resources. The code bellow provides one example on how to classify a small cube image provided with `sits` package.

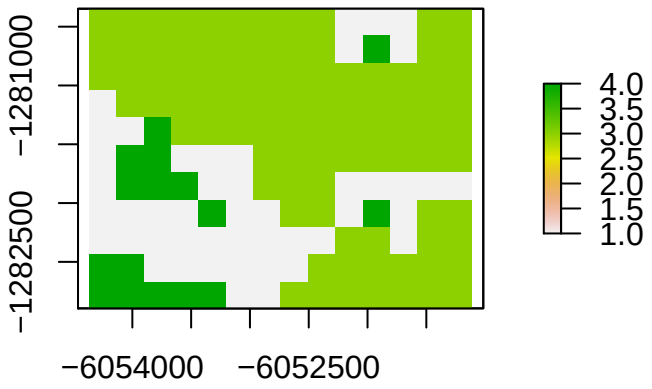
```
# Retrieve the set of samples for the Mato Grosso region
data(samples_MT_ndvi)

# read a raster file and put it into a vector
files  <-
    system.file("extdata/raster/mod13q1/sinop-crop-ndvi.tif",
                package = "sits")

# define the timeline
data(timeline_mod13q1)
timeline <- lubridate::as_date(timeline_mod13q1$V1)

# create a raster metadata file based on the
# information about the files
raster.tb <-
    sits_coverage(service = "RASTER",
                  product = "MOD13Q1",
                  name = "Sinop-crop",
                  timeline = timeline,
                  bands = c("ndvi"),
                  files = files)

# classify the raster file
raster_class.tb <-
    sits_classify_raster(file = "./raster-class",
                         raster.tb,
                         samples_MT_ndvi,
                         ml_method = sits_svm(),
                         blocksize = 300000,
                         multicores = 1)
# plot classified image
plot(raster_class.tb$r_obj[[1]])
```



Final remarks

Current approaches to image time series analysis still use limited number of attributes. A common approach is deriving a small set of phenological parameters from vegetation indices, like beginning, peak, and length of growing season [Brown et al., 2013], [Kastens et al., 2017], [Estel et al., 2015], [Pelletier et al., 2016]. These phenological parameters are then fed in specialised classifiers such as TIMESAT [Jönsson and Eklundh, 2004]. These approaches do not use the power of advanced statistical learning techniques to work on high-dimensional spaces and with big training data sets [James et al., 2013].

The `sits` uses the full depth of satellite image time series to create larger dimensional spaces. We tested different methods of extracting attributes from time series data, including those reported by Pelletier et al. [2016] and Kastens et al. [2017]. Our conclusion is that part of the information in raw time series is lost after filtering or statistical approximation. Thus, the method we developed has a deceptive simplicity: *use all the data available in the time series samples.* The idea is to have as many temporal attributes as possible, increasing the dimension of the classification space. Our experiments found out that modern statistical models such as support vector machines, and random forests perform better in high-dimensional spaces than in lower dimensional ones.

Acknowledgements

The authors would like to thank all the researchers whose provided data samples used in the examples: Alexandre Coutinho, Julio Esquerdo and Joao Antunes (Brazilian Agricultural Research Agency, Brazil) who provided ground samples for "soybean-fallow", "fallow-cotton", "soybean-cotton", "soybean-corn", "soybean-millet", "soybean-sunflower" and "pasture" classes; Rodrigo Bergotti (National Institute for Space Research, Brazil) who provided samples for "cerrado" and "forest" classes; and Damien Arvor (Rennes University, France) who provided ground samples for "soybean-fallow" class.

References

References

Agostino Accardo, M Affinito, M Carrozzi, and F Bouquet. Use of the fractal dimension for the analysis of electroencephalographic time series. *Biological cybernetics*, 77(5): 339–350, 1997.

Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

D. Arvor, M. Meirelles, V. Dubreuil, A. Bégué, and Y. E. Shimabukuro. Analyzing the agricultural transition in Mato Grosso, Brazil, using satellite-derived indices. *Applied Geography*, 32(2):702–713, 2012.

Peter M Atkinson, C Jeganathan, Jadu Dash, and Clement Atzberger. Inter-comparison of four models for smoothing satellite sensor time-series data to estimate vegetation phenology. *Remote sensing of environment*, 123:400–417, 2012.

Clement Atzberger and Paul HC Eilers. Evaluating the effectiveness of smoothing algorithms in the absence of ground reference measurements. *International Journal of Remote Sensing*, 32(13):3689–3709, 2011.

Bethany A Bradley, Robert W Jacob, John F Hermance, and John F Mustard. A curve fitting procedure to derive inter-annual phenologies from time series of noisy satellite NDVI data. *Remote Sensing of Environment*, 106(2):137–145, 2007.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

J. Christopher Brown, Jude H. Kastens, Alexandre Camargo Coutinho, Daniel de Castro Victoria, and Christopher R. Bishop. Classifying multiyear agricultural land use data from Mato Grosso using time-series MODIS vegetation index data. *Remote Sensing of Environment*, 130:39–50, 2013.

Gilberto Câmara, Luiz Fernando Assis, Gilberto Ribeiro, Karine Reis Ferreira, Eduardo Llapa, and Lubia Vinhas. Big earth observation data analytics: matching requirements to system architectures. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 1–6, Burlingname, CA, USA, 2016. ACM.

Gilberto Câmara, Gilberto Queiroz, Lubia Vinhas, Karine Ferreira, Ricardo Cartaxo, Rolf Simoes, Eduardo Llapa, Luiz Assis, and Alber Sanchez. The e-sensing architecture for big earth observation data analysis. In *Big Data from Space (BiDS'17)*, pages 48–51, 2017.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

Jin Chen, Per. Jönsson, Masayuki Tamura, Zhihui Gu andBunkei Matsushita, and Lars Eklundh. A simple method for reconstructing a high-quality NDVI time-series data set based on the Savitzky–Golay filter. *Remote Sensing of Environment*, 91(3-4):332 – 344, 2004.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20 (3):273–297, 1995.

EMBRAPA. Sistema de Análise Temporal da Vegetação (SATVEG), 2014. URL www.satveg.cnptia.embrapa.br.

Stephan Estel, Tobias Kuemmerle, Camilo Alcantara, Christian Levers, Alexander Prishchepov, and Patrick Hostert. Mapping farmland abandonment and recultivation across Europe using MODIS NDVI time series. *Remote Sensing of Environment*, 163: 312–325, 2015.

Gillian L Galford, John F Mustard, Jerry Melillo, Aline Gendrin, Carlos C Cerri, and Carlos E Cerri. 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, 2008.

Cristina Gomez, Joanne C. White, and Michael A. Wulder. Optical remotely sensed time series data for land cover classification: A review. *{ISPRS} Journal of Photogrammetry and Remote Sensing*, 116:55 – 72, 2016. ISSN 0924-2716.

M. C. Hansen, P. V. Potapov, R. Moore, M. Hancher, S. A. Turubanova, A. Tyukavina, D. Thau, S. V. Stehman, S. J. Goetz, T. R. Loveland, A. Kommareddy, A. Egorov, L. Chini, C. O. Justice, and J. R. G. Townshend. High-resolution global maps of 21st-century forest cover change. *Science*, 342(6160):850–853, 2013.

Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *IEEE transactions on pattern analysis and machine intelligence*, (4):532–550, 1987.

T. Hastie, R. Tibshirani, and Friedman J. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, New York, 2009.

Christian Hennig. Clustering strategy and method selection. In Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci, editors, *Handbook of cluster analysis*. CRC Press, 2015.

Robert J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2015.

Jennifer N Hird and Gregory J McDermid. Noise reduction of NDVI time series: An empirical comparison of selected techniques. *Remote Sensing of Environment*, 113(1): 248–258, 2009.

Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282, 1995.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2 (1):193–218, 1985.

INPE. Amazon Deforestation Monitoring Project (PRODES), 2017. URL www.obt.inpe.br/prodes.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, New York, EUA, 2013.

Per Jönsson and Lars Eklundh. TIMESAT—a program for analyzing time-series of satellite sensor data. *Computers & Geosciences*, 30(8):833 – 845, 2004. ISSN 0098-3004.

J. Kastens, J. Brown, A. Coutinho, C. Bishop, and J. Esquerdo. Soy moratorium impacts on soybean and deforestation dynamics in Mato Grosso, Brazil. *PLOS ONE*, 12(4): e0176168, 2017.

Robert E. Kennedy, Zhiqiang Yang, and Warren B. Cohen. Detecting trends in forest disturbance and recovery using yearly Landsat time series. *Remote Sensing of Environment*, 114(12):2897–2910, 2010.

Eamonn Keogh, Jessica Lin, and Wagner Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 115–122, 2003.

E.F. Lambin and M. Linderman. Time series of remote sensing data for land change science. *IEEE Transactions on Geoscience and Remote Sensing*, 44(7):1926–1928, 2006.

Eric F Lambin, Helmut J Geist, and Erika Lepers. Dynamics of land-use and land-cover change in tropical regions. *Annual review of environment and resources*, 28(1):205–241, 2003.

Hannibal H Madden. Comments on the Savitzky-Golay convolution method for least-squares-fit smoothing and differentiation of digital data. *Analytical chemistry*, 50(9): 1383–1386, 1978.

Valerie J. Pasquarella, Christopher E. Holden, Les Kaufman, and Curtis E. Woodcock. From imagery to ecology: leveraging time series of all available LANDSAT observations to map and monitor ecosystem state and dynamics. *Remote Sensing in Ecology and Conservation*, 2(3):152–170, 2016. ISSN 2056-3485. doi: 10.1002/rse2.24.

Charlotte Pelletier, Silvia Valero, Jordi Inglada, Nicolas Champion, and Gerard Dedieu. Assessing the robustness of Random Forests to map land cover with high resolution satellite image time series over large areas. *Remote Sensing of Environment*, 187: 156–168, 2016.

François Petitjean, Jordi Inglada, and Pierre Gançarskv. Clustering of satellite image time series under time warping. In *Analysis of Multi-temporal Remote Sensing Images (Multi-Temp), 2011 6th International Workshop on the*, pages 69–72. IEEE, 2011.

Gilberto Ribeiro de Queiroz, Karine Reis Ferreira, Lubia Vinhas, Gilberto Camara, Raphael Willian da Costa, Ricardo Cartaxo Modesto de Souza, Victor Wegner Maus, and Alber Sanchez. WTSS: um serviço web para extração de séries temporais de imagens de sensoriamento remoto. In *Proceeding of the XVII Remote Sensing Brazilian Symposium*, pages 7553–7560, 2015.

LAS Romani, RRV Gonçalves, BF Amaral, DYT Chino, J Zullo, C Traina, EPM Sousa, and AJM Traina. Clustering analysis applied to NDVI/NOAA multitemporal images to improve the monitoring process of sugarcane crops. In *Analysis of Multi-temporal Remote Sensing Images (Multi-Temp), 2011 6th International Workshop on the*, pages 33–36. IEEE, 2011.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Toshihiro Sakamoto, Masayuki Yokozawa, Hitoshi Toritani, Michio Shibayama, Naoki Ishitsuka, and Hiroyuki Ohno. A crop phenology detection method using time-series MODIS data. *Remote Sensing of Environment*, 96(3-4):366–374, 2005.

Yang Shao, Ross S. Lunetta, Brandon Wheeler, John S. Iiames, and James B. Campbell. An evaluation of time-series smoothing algorithms for land-cover classifications using MODIS-NDVI multi-temporal data. *Remote Sensing of Environment*, 174:258–265, 2016.

F. Vávra, P. Nový, H. Mašková, M. Kotlíková, and A. Netrvalová. Morphological filtration for time series. In *Proceedings of the 3rd International Conference APLIMAT*, pages 983–989, Bratislava, Slovakia, 2004. Slovak University of Technology in Bratislava.

Lubia Vinhas, Gilberto Ribeiro, Karine Reis Ferreira, and Gilberto Camara. Web Services for big Earth observation data. In *Proceedings of the 17th Brazilian Symposium on GeoInformatics*, pages 26–35, Campos do Jordão, SP, Brazil, 2016. INPE.

Joe H Ward. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

Edmund T Whittaker. On a new method of graduation. *Proceedings of the Edinburgh Mathematical Society*, 41:63–75, 1922.

Hadley Wickham and Garrett Grolemund. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc., 2017.

Achim Zeileis and Gabor Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27, 2005.

Jie Zhou, Li Jia, Massimo Menenti, and Ben Gorte. On the performance of remote sensing time series reconstruction methods – a spatial comparison. *Remote Sensing of Environment*, 187:367–384, 2016.