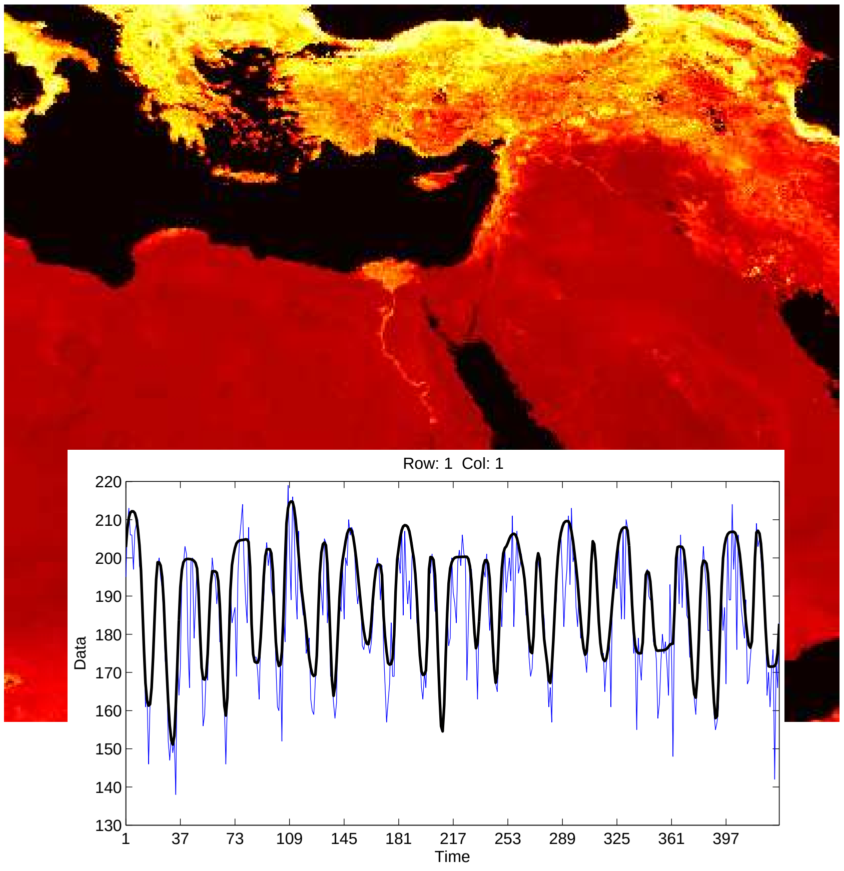
TIMESAT 3.3
with seasonal trend decomposition and parallel processing
Software Manual



Lars Eklundh[a] and Per Jönsson[b]

[a] Department of Physical Geography and Ecosystem Science, Lund University, Sweden
[b] Department of Materials Science and Applied Mathematics, Malmö University, Sweden
`http://www.nateko.lu.se/TIMESAT/`

2017-05-29

Contents

Part I
Introduction to TIMESAT

1 Introduction

Time-series of vegetation index derived from satellite spectral measurements can be used to gain information on seasonal vegetation development. This information aids analyzes of the functional and structural characteristics of the global and regional land cover and adds to our current knowledge of global cycles of energy and matter. Long time-series of vegetation index data can also provide information on shifts in the spatial distribution of bio-climatic zones, indicating variations in large-scale circulation patterns or land-use changes.

Although the value of remotely sensed time-series data for monitoring vegetation seasons has been firmly established, only a limited number of generally available methods exist for exploring and extracting seasonality parameters from such data series. For this reason the TIMESAT program package for extracting seasonal parameters has been developed.

1.1 About TIMESAT and the software manual

The TIMESAT 3.3 software manual consists of three parts. Part I gives general information together with examples of some applications of TIMESAT. Part II describes the algorithms underlying the software package. Also the settings affecting the processing are discussed in detail. Part III is the software user's guide, with detailed information on how to install, run, and handle the program package.

The TIMESAT program package is designed primarily for analyzing time-series of satellite data and uses an adaptive Savitzky-Golay filtering method and methods based on upper envelope weighted fits to asymmetric Gaussian and double logistic model functions (Jönsson and Eklundh, 2002, 2003, 2004). From the fitted model functions a number of seasonality parameters, e.g. beginning and end of the growing season, can be extracted. Parameters for a number of pixels can be merged into a map displaying seasonality on a regional or global scale.

TIMESAT consists of a number of numerical and graphical routines coded in Matlab and Fortran. TIMESAT is normally run from Matlab. However, also users without Matlab installed, or even without a Matlab license, can use the software. Processing is then done through a runtime engine called the Matlab Compiler Runtime (MCR), that is set up on the users machine by executing the file `MCRinstall.exe` (see section 7.2). The latter file is provided along with the TIMESAT program package. Fortran routines are highly vectorized and efficient for use with large data sets. The Fortran routines are pre-compiled for Windows and Linux.

TIMESAT has been developed over many years. During these years a number of new features have been added. Below are the main features of TIMESAT 3.3:

- Contains several smoothing methods for time-series data
- Includes seasonal trend decomposition by LOESS (STL)
- Contains several methods for detection of outliers
- Allows for weighting of data using quality information
- Allows for fitting to the upper envelope of the data

- Contains four methods for defining start and end of growing seasons
- Generates 13 seasonality parameters
- Individual processing for different land cover classes
- Versatile graphical user interface (GUI)
- Runs with or without Matlab installed
- Runs under Linux and Windows
- Allows for parallel processing on multicore computers to handle massive data sets

1.2 TIMESAT version 3.3 vs. version 3.2

There are some changes in TIMESAT version 3.3 compared to version 3.2. The most important of these changes are:

- Seasonal trend decomposition by STL added.
- Added two new methods for setting start of season (SOS) and end of season (EOS): relative amplitude and STL trendline crossing.
- Added two new seasonality parameters: data value at SOS and data value at EOS.
- Changed the settings file format.
- Added several checks on seasonality consistency.
- Relaxed checks on no. of missing values in input data.
- Legend added in `TSM_GUI`; changed checkboxes for fitting methods into list. Last browse folder now remembered.
- Updated `TSF_seas2image`. The method now looks for a season based on the time for the maximum rather than SOS/EOS. This results in fewer missing data points.
- Fixed bug in parallel processing of large files.
- `NaN` data now allowed in image input files.
- Various cosmetic fixes.

For backward compatibility some obsolete routines are kept.

1.3 TIMESAT home page

On the TIMESAT home page `http://www.nateko.lu.se/TIMESAT/timesat.asp` the full program package, along with documentation and test data sets, is available for download in the form of a zip-file. The installation of the package from the zip-file is described in section 7.2 in Part III of this manual. **On the home page there are also answers to frequently asked questions as well as other information.**

1.4 Using and citing TIMESAT

Timesat is provided to the scientific community for non-commercial and non-military purposes. However, the intellectual ownership and the right to distribute the program remains with the creators. Access to the software is granted upon registration. Users of Timesat should quote this document and our two main publications:

Eklundh, L. and Jönsson, P., 2017, Timesat 3.3 Software Manual, Lund and Malmö University, Sweden.

Jönsson, P. and Eklundh, L., 2002, Seasonality extraction and noise removal by function fitting to time-series of satellite sensor data, IEEE Transactions of Geoscience and Remote Sensing, 40, No 8, 1824 – 1832.

Jönsson, P. and Eklundh, L., 2004, Timesat - a program for analyzing time-series of satellite sensor data, Computers and Geosciences, 30, 833 – 845.

Timesat is the intellectual property of Per Jönsson, Malmö University, Sweden and Lars Eklundh, Lund University, Sweden. In case of questions, suggestions or any comments please contact the authors by e-mail at `per.jonsson@mah.se` and `lars.eklundh@nateko.lu.se`. We cannot guarantee user support, but we will do our best to answer questions.

1.5 Applications of TIMESAT

Timesat has been used in a number of applications, e.g. for characterizing phenology (Eklundh and Jönsson 2003) and for mapping environmental and phenological changes in Africa from 1982 till today (Eklundh and Olsson 2003, Olsson *et al.* 2005, Seaquist *et al.* 2006, Heumann *et al.* 2007, Hickler *et al.* 2005, Seaquist *et al.* 2009), for improving data in ecosystem classification (Tottrup *et al.* 2007), for use with MSG SEVIRI data (Stisen *et al.* 2007), for mapping high-latitude forest phenology (Beck *et al.* 2007), and to evaluate satellite and climate data-derived indices of fire risk in savanna ecosystems (Verbesselt *et al.* 2006) as well as to monitor human footprints on fire seasons (Le Page *et al.* 2010).

We use Timesat as an integrated part in our development of carbon models based on data from Terra/MODIS (Olofsson and Eklundh 2007, Olofsson *et al.* 2007, 2008, Sjöström *et al.* 2009, 2011, Schubert et al. 2010, 2012), and for analyzing relationships between NDVI of nemoboreal and boreal coniferous forests and models of conifer cold hardiness, bud burst and photosynthetic efficiency (Jönsson et al. 2010). We also use Timesat with Terra/MODIS data in the development of systems for detection of forest disturbances, e.g. due to insect infestations (Eklundh *et al.* 2009).

A modified version of Timesat v. 2.3 is integrated in the processing of MODIS data into a phenology product (MOD09PHN and MOD15PHN) by the North American Carbon Program (Gao *et al.* 2008).

TIMESAT has also been used for improving the data quality of MODIS and AVHRR satellite products (Yuan *et al.* 2011, Fensholt and Proud 2012, Barichivich *et al.* 2013).

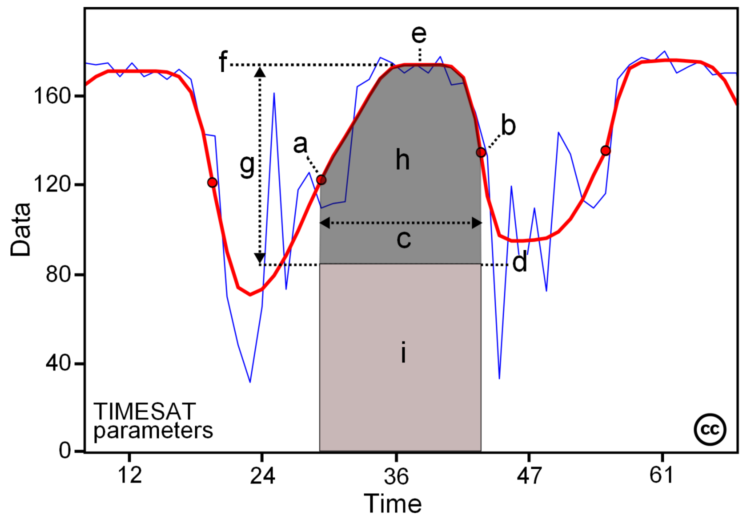
A more complete summary of applications of Timesat is given in L. Eklundh and P. Jönsson. "Timesat: A software package for time-series processing and assessment of vegetation dynamics," in *Remote Sensing Time Series*, C. Kuenzer, S. Dech and W. Wagner, Ed. Heidelberg: Springer, 2015, pp. 141-158.

1.6 About the authors

Lars Eklundh received the Ph.D. degree in physical geography from Lund University, Lund, Sweden in 1996. He is currently Professor at Lund University. He was with the United Nations Environment Program (UNEP) from 1989 to 1992. His primary research interest is remote sensing for the analysis of spatial and temporal variation of vegetation parameters. Main fields of application include climate variability, carbon cycle research, phenology, and forest disturbances. He is funded by the Swedish National Space Board and the Swedish Research Council FORMAS.

Per Jönsson studied mathematics, physics, and astronomy and received the Ph.D. degree in physics from Lund University, Lund, Sweden in 1995. From 1995 to 1997 he was a Post Doctoral Research Assistant in computer science at Vanderbilt University, Nashville, USA. Per Jönsson holds a position as Professor in applied mathematics at Malmö University, Sweden. His primary research interest is in computational science with applications to atomic- plasma-, and astrophysics. More recently he has geared into remote sensing and ecosystem analysis. Per is a partner in the program for remote sensing and phenology modeling for detecting forest changes due to climate change. His research in remote sensing is supported by the Swedish National Space Board.

Part II
Algorithm Theoretical Basis Document



Some of the seasonality parameters generated in TIMESAT: (a) beginning of season, (b) end of season, (c) length of season, (d) base value, (e) time of middle of season, (f) maximum value, (g) amplitude, (h) small integrated value, (h+i) large integrated value. This figure is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 2.5 Sweden License. It is free to copy and use in other work.

2 Overview of data processing

TIMESAT is primarily designed to process time-series of vegetation index derived from satellite spectral measurements. However, other types of data such as meteorological index, fire data, and eddy co-variance carbon flux data can also be processed (Verbesselt *et al.*, 2006, Le Page *et al.*, 2010). Ancillary quality data may also be used to guide the processing of the time-series. Sequential data as well as data organized in images (two-dimensional spatial arrays) can be handled.

2.1 Sequential data

In the simplest setting TIMESAT processes a number of time-series in an ASCII file. The first line of the ASCII file gives information about the number of years spanned by the time-series, *nyear*, the number of data values per year, *nptperyear*, and the number of time-series in the file, *nts*. Below the first line the time-series $y_1, y_2, \ldots, y_N$, with $N = nyear \times nptperyear$, are given line by line. In total there are *nts* lines with time-series. Optionally, the time-series file is accompanied with a file in the same format with quality indicators $q_1, q_2, \ldots, q_N$. Given the number of data values per year, the indices $1, 2, \ldots, N$ of the time-series directly translates to time values $t_1, t_2, \ldots, t_N$. The data structure of the ASCII file is displayed in Figure 1.

$$
\begin{array}{l}
nyear\ nptperyear\ nts \\
\left.\begin{array}{llll}
y_1 & y_2 & \cdots & y_N \\
y_1 & y_2 & \cdots & y_N \\
 & & & \\
y_1 & y_2 & \cdots & y_N
\end{array}\right\} nts
\end{array}
$$

Figure 1: *Data structure of ASCII files containing time-series. The first line of the file gives information about the number of years spanned by the time-series, nyear, the number of data values per year, nptperyear, and the number of time-series in the file, nts. After the first line with general data the time-series are given line by line.*

Given the ASCII file with time-series and, optionally, a file with quality indicators and a file with input settings (see sections 5.2 and 5.3), TIMESAT fits a smooth function to each of the time-series and extracts seasonality data such as start and end of the season or length of season. The steps of the processing are summarized below.

1. Read input settings that define the processing of the time-series.
2. Read first line of the ASCII files giving information about the length and sampling of a time-series as well as the number of time-series.
3. Loop over the time-series in the file.
   (a) Read time-series $y_1, y_2, \ldots, y_N$ and, optionally, quality indicators $q_1, q_2, \ldots, q_N$.
   (b) Pre-process time-series under the guidance of the quality indicators.

(c) Fit a smooth function to the time-series.

(d) Use fitted function to extract seasonality parameters.

(e) Write seasonality parameters and fitted function to file.

The original time-series and the fitted functions can be displayed by the TIMESAT routines or by user programs written in e.g. Matlab. In Figure 2 we display a processed time-series covering five years. The start and end of the seasons are marked with filled circles.

Settings defining the regression model for the fits, how outliers and quality data are handled etc. are given in the input settings file. For convenience the settings are summarized in sections 5.2 and 5.3 of this document. The input settings file can be created manually or, better, by using the TIMESAT graphical user interface (GUI). The user interface is described in Part III of this manual.

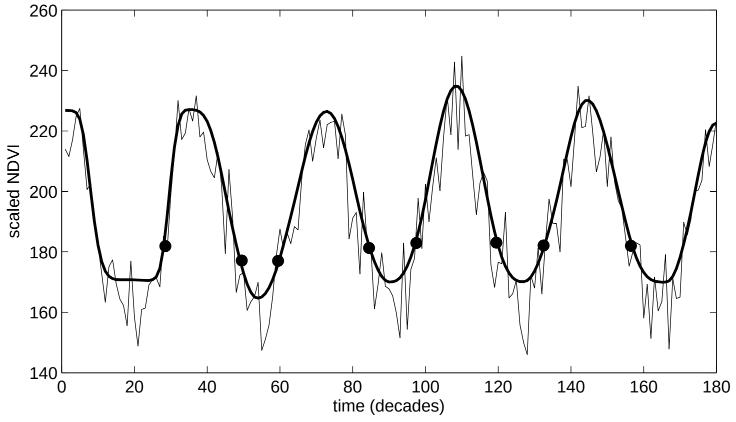


Figure 2: *Time-series $y_1, y_2, \ldots, y_N$ covering a period of 5 years. Seasonality parameters from the 4 full seasons are determined from the fitted functions.*

2.2 Image data

Remotely sensed data are often organized in binary images (two-dimensional spatial arrays). Though any kind of remotely sensed data can be used, vegetation index data are very common in time-series analysis. We will thus refer to time-series image data as vegetation index data in the following text. Each image gives the vegetation index values in the array at a specified time. By extracting vegetation index values at a pixel (j, k) in the array for consecutive times, a time-series $y_1, y_2, \ldots, y_N$ is obtained for this pixel (see Figure 3). In some cases data are complemented by quality data that are organized in images in a similar way.

Frequently data have been spatially clustered and each pixel (j, k) has been assigned a land cover class e.g. water, cropland, deciduous, and coniferous forest. Time-series in a class share some characteristics and are sometimes quite different from time-series in other classes. It is advantageous to be able to use separate processing schemes for each class.

Given a stack of images with vegetation index and, optionally, a stack of images with quality indicators, and a file with input settings (see sections 5.2 and 5.3), TIMESAT smooths each

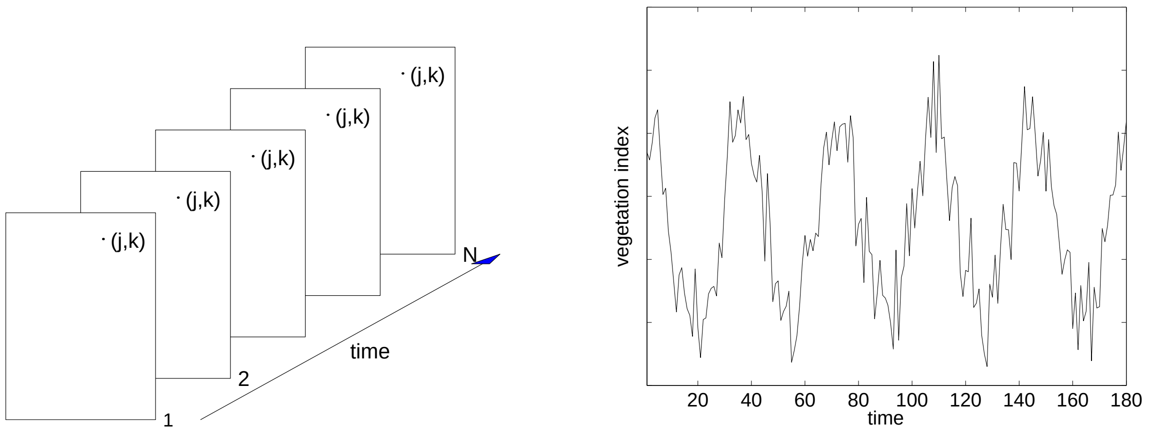


Figure 3: *Vegetation index data are organized in images (left panel). An image i gives index values over an area for the time t_i. By extracting values at a pixel (j, k) for consecutive times, a time-series $y_1, y_2, \ldots, y_N$ is obtained for this pixel (right panel). The pixel may be assigned a land use class, and it is possible to use separate processing schemes for each class.*

of the time-series in a defined area and extracts seasonality data such as start and end of the season or the length of the season. The details of the processing, which depends on the assigned land use class of the pixel, are summarized below.

1. Read file containing the names of all the images with vegetation data, and optionally, the images with quality data. Read file with land use classes.

2. Give spatial extension of the area in the image that should be processed. Supply information about the image format (8-bit unsigned integer, 16-bit signed integer etc).

3. Read general as well as land use class-specific input settings that define the processing of the time-series.

4. Loop over pixels (j, k), serial or in parallel, in the defined area. For each pixel:

   (a) Extract time-series $y_1, y_2, \ldots, y_N$ and, optionally, quality indicators $q_1, q_2, \ldots, q_N$ from images.

   (b) Read land use classification for the current pixel.

   (c) Pre-process time-series under the guidance of the quality indicators and land use classification.

   (d) Fit a smooth function to the time-series.

   (e) Use fitted function to extract seasonality parameters.

   (f) Write seasonality parameters and fitted function to files.

5. Read files with seasonality parameters and generate maps.

Files with extracted parameters are used to generate seasonality maps or images at regional or continental scale. Comparing images of the same quantity for consecutive years may reveal

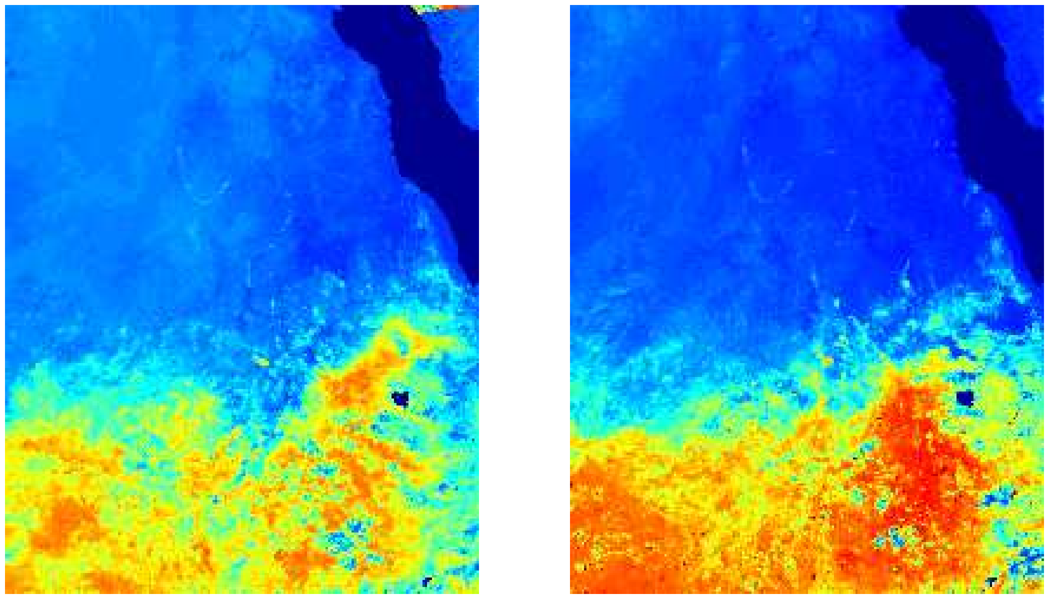


Figure 4: *Seasonal amplitude in Ethiopia in 1982 and 2000 from functions fitted to data from NOAA/AVHRR. Red is translated into high amplitude in the image and it is seen that there are some distinct changes in the vegetation index between the years.*

shifts in vegetation coverage due to climate change or other dynamical events (Eklundh and Olsson 2003, Heumann *et al.* 2007). In Figure 4, two images of the seasonal amplitude, as derived from functions fitted to data from NOAA/AVHRR, in an area covering a portion of N. Eastern Africa are displayed. The image to the left displays the amplitude in 1982 and the image to the right displays the amplitude in 2000.

3 Methodology

Timesat implements three processing methods based on least-squares fits to the upper envelope of the vegetation index data. The first method uses local polynomial functions in the fitting, and the method can be classified as an adaptive Savitzky-Golay filter. The other two methods are least-squares methods, where data are fitted to non-linear model functions of different complexity. All three processing methods use a preliminary definition of the seasonality (uni-modal or bi-modal) along with approximate timings of the growing seasons. In addition Timesat 3.3 implements Seasonal Trend Decomposition by LOESS (STL) (Cleveland et al. 1990).

We start by a general description of weighted least-squares fits. Pre-processing and removal of outliers are discussed, and then we go on to describe an iterative method to adapt the fitted functions to the upper envelope of the data. This is followed by an account on how to determine the number of annual growing seasons and their approximate timing. The details of the processing methods are given, and finally the extraction of seasonality information is described.

3.1 Least-squares fitting

Assume that we have a time-series (t_i, y_i), $i = 1, 2, \ldots, N$ and a model function $f(t)$ of the form

$$f(t) = c_1\varphi_1(t) + c_2\varphi_2(t) + \ldots + c_M\varphi_M(t), \tag{1}$$

where $\varphi_1(t), \varphi_2(t), \ldots, \varphi_M(t)$ are given basis functions. The best values, in the least-squares sense, of the parameters $c_1, c_2, \ldots, c_M$ are obtained by solving the system of normal equations

$$\mathbf{A}^T\mathbf{A}\mathbf{c} = \mathbf{A}^T\mathbf{b}, \tag{2}$$

where

$$A_{ij} = w_i\varphi_j(t_i), \qquad b_i = w_i y_i. \tag{3}$$

Here w_i is the weight of the ith data value, presumed to be known. Values with small weights will influence the fit less than values with large weights. If the weights are not known they may all be set to the constant value $w = 1$ (Press *et al.* 1994).

3.2 On the use of ancillary quality data for assigning weights

In Timesat cloud classifications and other ancillary data may be used to assign weights to the values in the time-series, such as the QA quality labels accompanying the MODIS satellite sensor data. Another example, used in the Timesat tutorial, is the Pathfinder AVHRR Land (PAL) Normalized Difference Vegetation Index (NDVI) accompanied by Clouds from AVHRR (CLAVR) that is a cloud indicator based on thresholds of the AVHRR reflectance and thermal channels. The original CLAVR data lie between 1 and 31, representing the three broad groups; cloudy (1 – 11), mixed (12 – 21), and clear (22 – 31). Values in the time-series associated with these three groups can be assigned different weights. In previous work weights $w = 0$, 0.5 and 1 have been used for values in the time-series associated with, respectively, cloudy, mixed and clear conditions. There are, of course, no general rules for converting ancillary data to weights associated with the values in the time-series, and the user of the Timesat program is encouraged to take an experimental approach and test different settings. Figure 5(a) depicts a time-series for which the values have been assigned weights based on the CLAVR values. Large circles indicate clear conditions ($w = 1$), small circles indicate mixed conditions ($w = 0.5$), and no circle indicate clouds ($w = 0$). From the figure it is seen that several of the negatively biased outliers are associated with cloudy conditions. By assigning zero weight to these cloudy values they will not influence the subsequent fitting.

3.3 Pre-processing to remove spikes and outliers

As we have seen some spikes and outliers may be detected using ancillary quality data. In many time-series there are, however, remaining positive and negative outliers that seriously impair the function fits. Three different methods for removing these outliers can be selected. In the first method a data value is defined as an outlier following two criteria: (1) it deviates more than a maximum deviation (here called cutoff) from the median in a moving window (half window width = number of values per year/7), and (2) it is lower than the mean value of its immediate neighbors minus the cutoff ($y(t) < \text{mean}(y(t-1), y(t+1)) - \text{cutoff}$) or it is larger than the highest value of its immediate neighbor plus the cutoff ($y(t) > \max(y(t -$

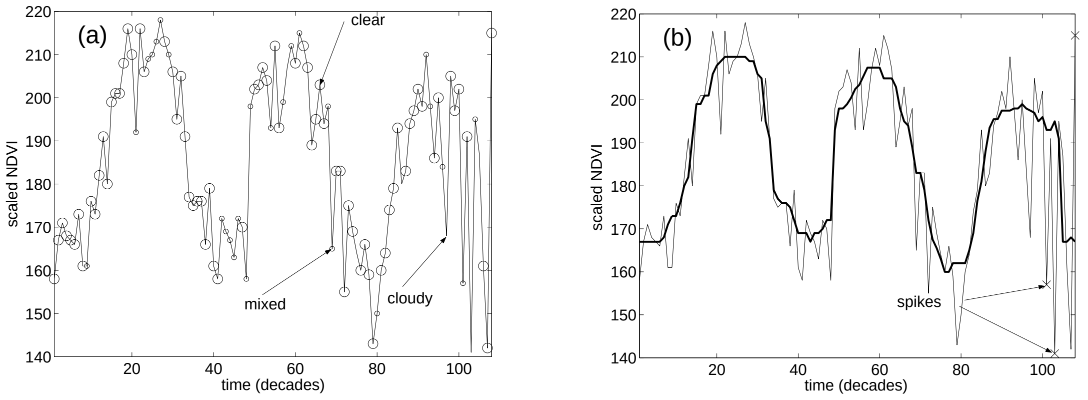


Figure 5: *(a) Time-series where the values have been assigned weights:* $w = 1$ *(large circles),* $w = 0.5$ *(small circles), and* $w = 0$ *(no circle). (b) Time-series together with values from a median filtering. Values in the time-series that are sufficiently different from both the left- and right-hand neighbors and the median filtered value are classified as outliers and are assigned weight 0. Detected spikes (outliers) are marked by crosses.*

$1), y(t+1)) + \text{cutoff})$. The cutoff is the standard deviation of the entire time-series times a factor given by the user. In the second method, which is more global in character and not dependent on ancillary data, values in the time-series are assigned weights based on an STL-decomposition (Cleveland *et al.* 1990). STL is described in section 3.10. Finally, in the third method values in the time-series are assigned weights that are products of the weights from the STL-decomposition and weights assigned based on the ancillary data. It is important to pay attention to the pre-processing, since remaining spikes and outliers may seriously degrade the final function fits. Since pre-processing is data dependent we recommend the user to take an experimental approach and test different settings.

3.4 Adaption to the upper envelope

To take into account that most noise in NDVI and other vegetation indices generated from remotely sensed land data is negatively biased, the determination of the parameters $c_1, c_2, \ldots, c_M$ of the model function is done in two or more steps. In the first step the parameters are obtained by solving the system of normal equations with weight $w_1, w_2, \ldots, w_N$ obtained from the ancillary cloud data or from the STL-decomposition. Data values below the model function of the first fit are thought of as being less important, and in the second step the system is solved with the weights of the low data values decreased by some factor. In TIMESAT this can be repeated 2 times. This multi-step procedure leads to a model function that is adapted to the upper envelope of the data (see Figure 6).

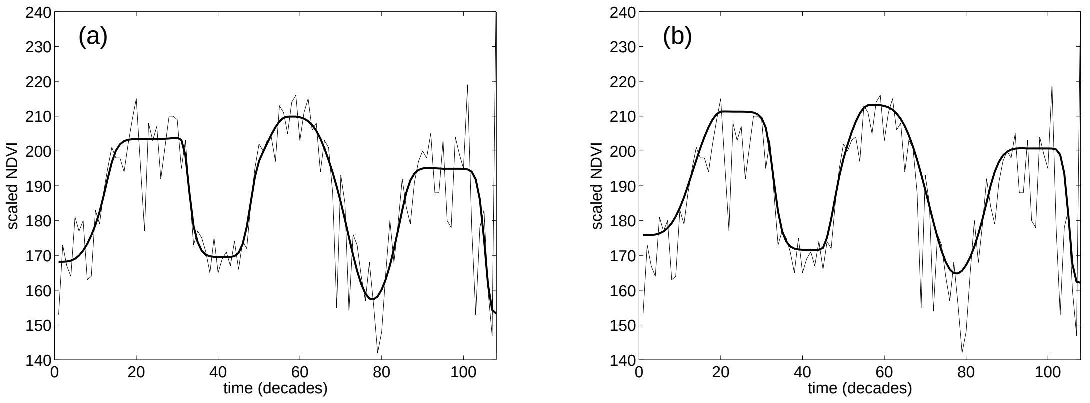


Figure 6: *Fitted functions from a multi-step procedure. The thin solid line represent the original NDVI data. (a) The thick line shows the fitted function from the first step. (b) The thick solid line displays the fit from the last step where the weights of the low data values have been decreased.*

3.5 Determination of the number of seasons

The high level of noise often makes it difficult to determine the number of annual seasons based on data for only one year. Including data from surrounding years reduces the risk for erroneous determinations. In TIMESAT, de-trended data values (t_i, y_i), $i = 1, 2, \ldots, N$ in the time-series are fit to a model function

$$f(t) = c_1 + c_2 \sin(\omega t) + c_3 \cos(\omega t) + c_4 \sin(2\omega t) + c_5 \cos(2\omega t), \tag{4}$$

where $\omega = 2\pi \cdot nyear/N$. The first basis function determines the base level whereas the pairs of sine and cosine functions correspond to, respectively, one and two annual vegetation seasons.

The fitting procedure always gives a primary maximum. In addition, a secondary maximum may be found. If the amplitude ratio between the secondary maximum and the primary maximum exceeds a user defined threshold – the seasonality parameter – we have two annual seasons. If the amplitude ratio is below the threshold we have one annual season (see figure 7). By carefully selecting the seasonality parameter TIMESAT will discriminate between noise and a second annual season. Setting the seasonality parameter to 1 forces the program to treat

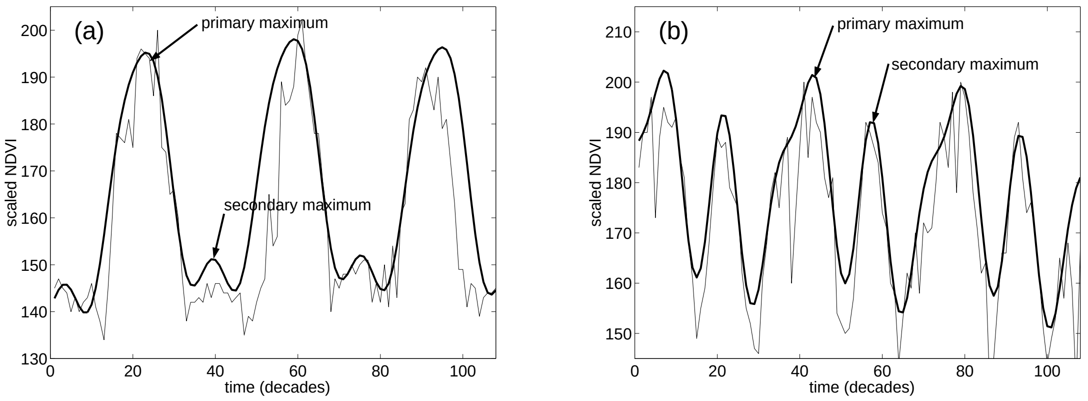


Figure 7: *If the amplitude ratio is below a user defined threshold we have one annual season. If the ratio is above the threshold we have two annual seasons.*

data as if there is one annual season. Setting the seasonality parameter to 0 forces the program to treat data as if there are two annual seasons. Information on the number of annual seasons is used further on in the TIMESAT program to define intervals in which to perform the local fits to Gaussians and double logistic functions (see section 3.9).

3.6 Adaptive Savitzky-Golay filtering

One way of smoothing data and suppressing disturbances is to use a filter, and replace each data value y_i, $i = 1, \ldots, N$ by a linear combination of nearby values in a window

$$\sum_{j=-n}^{n} c_j y_{i+j}. \tag{5}$$

In the simplest case, referred to as a moving average, the weights are $c_j = 1/(2n+1)$, and the data value y_i is replaced by the average of the values in the window. The moving average method preserves the area and mean position of a seasonal peak, but alters both the width and height. The latter properties can be preserved by approximating the underlying data value, not by the average in the window, but with the value obtained from a least-squares fit to a polynomial. For each data value $y_i,\ i = 1, 2, \ldots, N$ we fit a quadratic polynomial $f(t) = c_1 + c_2 t + c_3 t^2$ to all $2n+1$ points in the moving window and replace the value y_i with the value of the polynomial at position t_i. The procedure above is commonly referred to as a Savitzky-Golay filter (Press *et al.* 1994). To account for the negatively biased noise, the fitting is done in multiple steps as described in the previous section. The result is a smoothed curve adapted to the upper envelope of the values in the time-series.

The width, n, of the moving window determines the degree of smoothing, but it also affects the ability to follow a rapid change. In TIMESAT the width n can be set by the user. Even if the global setting of the moving window works fairly well, it is sometimes necessary to locally tighten the window. To capture the corresponding sudden rise in data values, only a small window can be used. In the program, the Savitzky-Golay filtering is performed using the global value n of the window. The filtered data are then scanned. If there is a large increase or decrease in an interval around a data point y_i, this data point will be associated with a smaller window. The filtering is then redone with the new locally adapted size of the window. The adaptive procedure, which is unique to TIMESAT, is illustrated in Fig. 8.

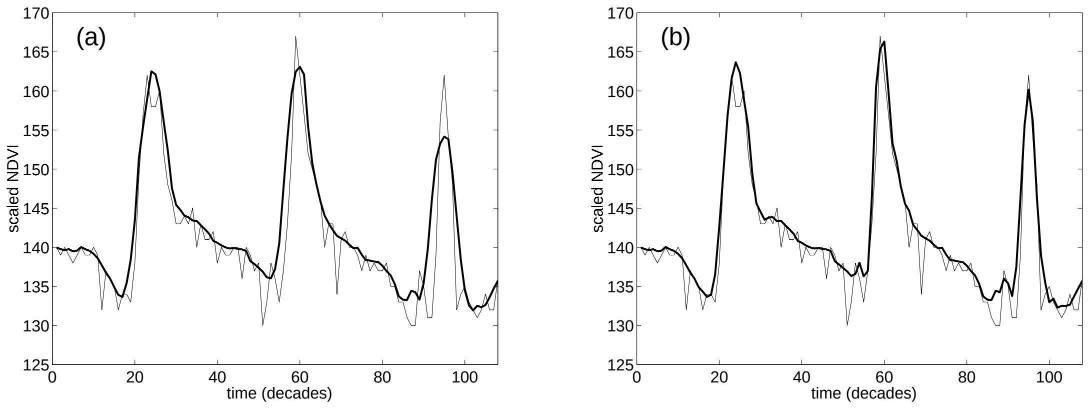


Figure 8: *In (a) the filtering is done with $n = 5$, which is too large for the filtered data to follow the sudden increase and decrease of the underlying data values. A scan of the filtered data identifies the data points for which there are large increases or decreases in surrounding intervals. Setting $n = 3$ for these points and redoing the filtering gives the curve in (b). Note the improved fit at the rising edges and at the narrow seasonal peaks.*

3.7 Fits to asymmetric Gaussians and double logistic functions

In these two methods local model functions are fit to data in intervals around maxima and minima in the time-series. The local model functions have the general form

$$f(t) \equiv f(t; \mathbf{c}, \mathbf{x}) = c_1 + c_2\, g(t; \mathbf{x}), \tag{6}$$

where the linear parameters $\mathbf{c} = (c_1, c_2)$ determine the base level and the amplitude. The non-linear parameters $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ determine the shape of the basis function $g(t; \mathbf{x})$. To make sure that the function space spanned by the basis function is physically reasonable in terms of how fast the function can grow or decline etc. the non-linear parameters $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ are restricted in range.

Asymmetric Gaussians

In this case the basis function

$$g(t;x_1,x_2,\ldots,x_5) = \begin{cases} \exp\left[-\left(\dfrac{t-x_1}{x_2}\right)^{x_3}\right] & \text{if } \ t > x_1 \\[2ex] \exp\left[-\left(\dfrac{x_1-t}{x_4}\right)^{x_5}\right] & \text{if } \ t < x_1 \end{cases} \tag{7}$$

is a Gaussian type of function. For this function x_1 determines the position of the maximum or minimum with respect to the independent time variable t, while x_2 and x_3 determine the width and flatness (kurtosis) of the right function half. Similarly, x_4 and x_5 determine the width and flatness of the left half. The effects of varying the parameters $x_2, \ldots, x_5$ are shown in Figure 9.

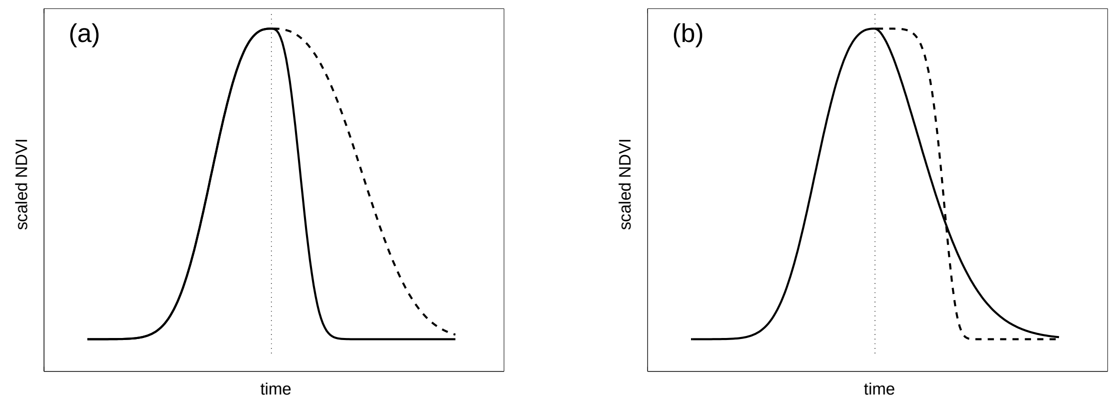


Figure 9: *Effect of parameter changes on the local functions. In (a) the parameter x_2, which determines the width of the right function half, has been decreased (solid line) and increased (dashed line) compared to the value of the left half. In (b) the parameter x_3, which determines the flatness of the right function half, has been decreased (solid line) and increased (dashed line) compared to the value of the left half.*

In order to ensure smooth shapes of the model functions, consistent with what is observed for data, the parameters $x_2, \ldots, x_5$ are restricted in range. For example, x_3 and x_5 are assumed to be larger than 2 in order to avoid a cusp at the matching point $t = x_1$ of the Gaussian function.

Double logistic functions

Here the basis function

$$g(t;x_1,...,x_4) = \frac{1}{1+\exp\left(\dfrac{x_1-t}{x_2}\right)} - \frac{1}{1+\exp\left(\dfrac{x_3-t}{x_4}\right)} \tag{8}$$

is a double logistic function. x_1 determines the position of the left inflection point while x_2 gives the rate of change. Similarly x_3 determines the position of the right inflection point while x_4 gives the rate of change at this point. Also for this function the parameters are restricted in range to ensure a smooth shape.

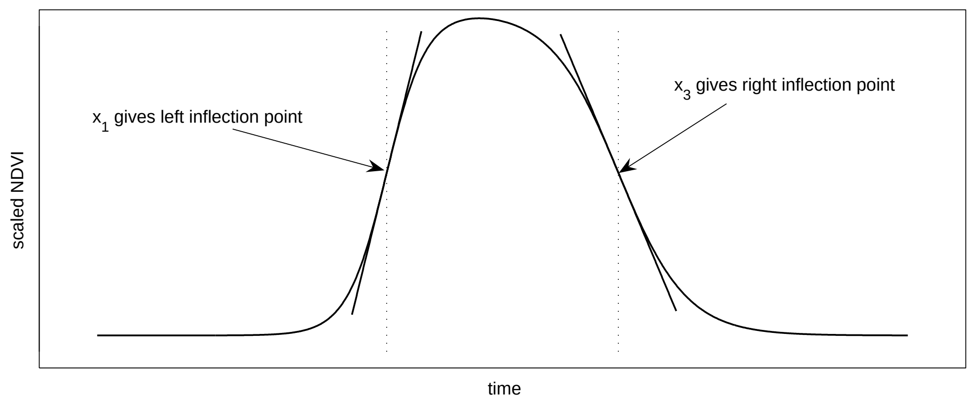


Figure 10: *In the double logistic function x_1 determines the position of the left inflection point while x_2 gives the rate of change. Similarly x_3 determines the position of the right inflection point while x_4 gives the rate of change at this point.*

3.8 Separable non-linear least-squares fits

The asymmetric Gaussian and double logistic model functions are well suited for describing the shape of the time-series in overlapping intervals around maxima and minima. Given a set of data points (t_i, y_i), $i = n_1, \ldots, n_2$ in an interval around a maximum or a minimum, the parameters $\mathbf{c}$ and $\mathbf{x}$ are obtained by minimizing the merit function

$$\chi^2 = \sum_{i=n_1}^{n_2} \left[w_i (f(t_i, \mathbf{c}, \mathbf{x}) - y_i) \right]^2 . \tag{9}$$

The function depends linearly on $\mathbf{c}$ and non-linearly on $\mathbf{x}$. In TIMESAT the minimization is done using a separable Levenberg-Marquardt method (Madsen *et al.* 2002; Nielsen 1999, 2000), where the box constraints on the non-linear parameters are enforced by projecting onto the feasible parameter interval (Kanzow *et al.* 2002). Initial values of the non-linear parameters are obtained by looping through a number of pre-defined model functions in a highly efficient search routine. The fitting is done in steps, as described in section 3.4, to account for the negatively biased noise.

3.9 Merging of local functions

The local model functions describe sensor data very well in broad intervals around maxima and minima (the location and length of the intervals depend on whether we have one or two annual seasons, see section 3.5). At the limbs, however, the fits are less good. Denoting the local functions describing the time-series in intervals around the left minimum, the central maximum and the right minimum by $f_L(t)$, $f_C(t)$, and $f_R(t)$ (see Figures 11(a-c)), the global function $F(t)$, that correctly models the time-series in the full interval $[t_L, t_R]$, is

$$F(t) = \begin{cases} \alpha(t) f_L(t) + [1 - \alpha(t)] f_C(t), & t_L < t < t_C \\ \beta(t) f_C(t) + [1 - \beta(t)] f_R(t), & t_C < t < t_R. \end{cases} \tag{10}$$

Here $\alpha(t)$ and $\beta(t)$ are cut-off functions that in small intervals around $(t_L + t_C)/2$ and $(t_C + t_R)/2$, respectively, smoothly drop from 1 to 0. Loosely speaking, the global function $F(t)$, shown in Figure 11(d), assumes the character of $f_L(t)$, $f_C(t)$ and $f_R(t)$ in, respectively, the left, central and right part of the interval $[t_L, t_R]$. The merging of local functions to a global function is a key feature of the method. It increases the flexibility and allows the fitted function to follow a complex behavior of the time-series (Jönsson and Eklundh, 2002).

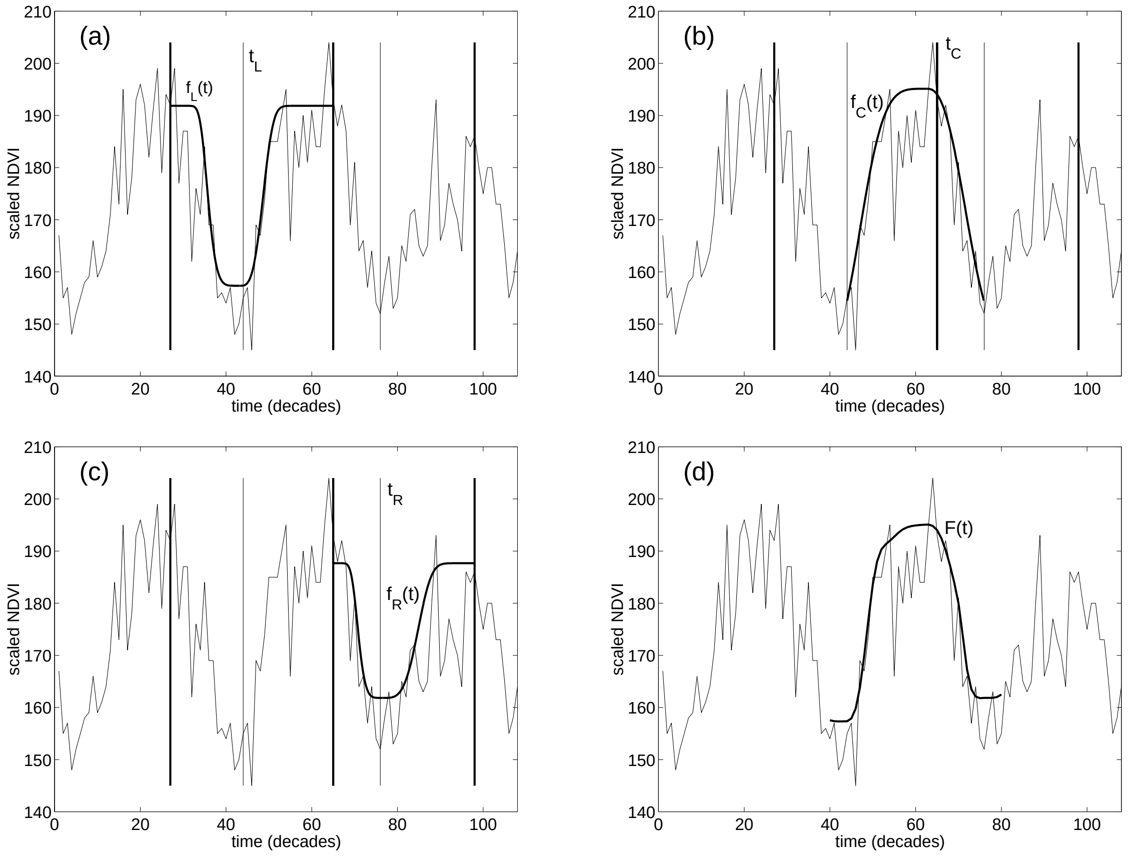


Figure 11: *(a-c) display local model functions fitted to, respectively, the left minimum, the central maximum, and the right minimum. (d) shows the global model function that is obtained by merging the three local functions.*

3.10 Seasonal trend decomposition

Trend decomposition is implemented in the STL method (Seasonal Trend decomposition by LOESS, Cleveland et al. 1990). This is a method based on a locally weighted regression smoother (LOESS), generating an output consisting of a non-linear trend line, a seasonal component, and a remainder (residual). In TIMESAT it is possible to generate the trend and the seasonal component (Figure 12) and write these to output files. The remainder is used in the outlier detection used for pre-processing (section 3.3). The trend component can be used as input in trend segmentation (Verbesselt et al. 2010, Jamali et al. 2015), and the seasonal component provides a generalized seasonal profile for the time-series. In TIMESAT it is possible to extract all seasonality parameters for the STL seasonal component, however, note that these are generally less accurate than those obtained from Savitzky-Golay filtering, asymmetric Gaussian or double-logistic functions.

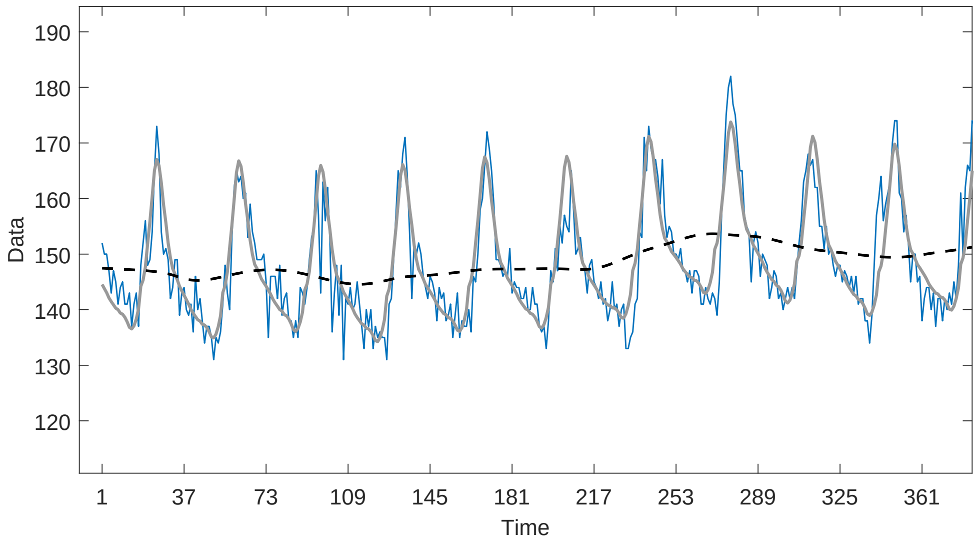


Figure 12: *Time series decomposed into seasonal component (gray solid line), and trend (dashed line).*

4 Extraction of seasonality parameters

Phenology is the response of the vegetation to seasonal climatic cycles in irradiance, temperature and rainfall. Therefore phenology constitutes an essential land surface parameter in atmospheric and climate models. However, the seasonal patterns observed in satellite-derived time-series data may also be affected by other cyclic and non-cyclic effects. Hence, we will use the term seasonality when describing these annually occurring events. Furthermore, seasonality parameters obtained from satellite derived time-series are often affected by the high degree of noise in the data. Using fitted functions reduces the uncertainties and leads to more stable measures.

4.1 Seasonality parameters derived from time-series spanning n years

Consider a time-series with one growing season per year. During a period of n years there may, in the general case, be $n-1$ full seasons together with two fractions of a season in the beginning and end of the time-series. Using the fitted functions seasonality parameters can be extracted for each of the $n-1$ full seasons (see Figure 13). If there are n seasons that each peaks in the middle of a year it would in principle be possible to extract seasonality parameters from each of the years (see for example Figure 5b). However, to be generally applicable, TIMESAT is not programmed for this, but will extract seasonality data only for the $n-1$ center-most seasons! To overcome this the user may add one year of dummy data in the beginning and one at the end of the time-series (see section below).

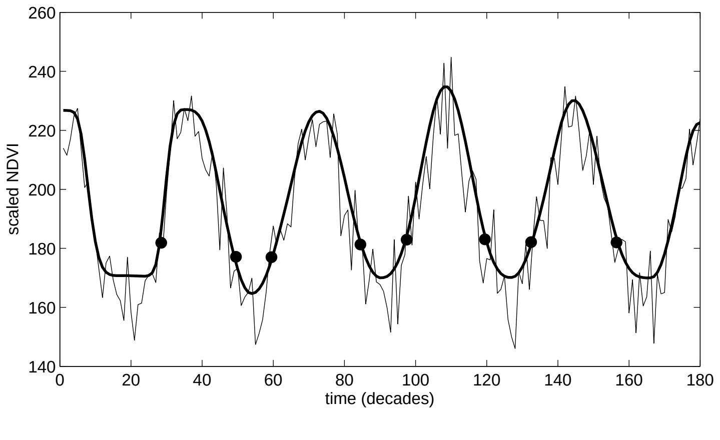


Figure 13: *Time-series covering a period of 5 years. Only seasonality parameters from the 4 full seasons can be determined from fitted functions. The start and end of the seasons are marked with filled circles.*

4.2 Extracting seasonality parameters from one year of data

If the vegetation season peaks around the middle of the time-series it is, in principle, possible to extract seasonality parameters from only one year of data. As discussed above, TIMESAT does not handle this case. To overcome this problem the user can duplicate the time-series and make an artificial time-series spanning three years (see Figure 14). The seasonality parameters extracted from the middle season of this artificial time-series are the desired ones. Note that this trick can only be used when the season peaks in the middle of the time-series.

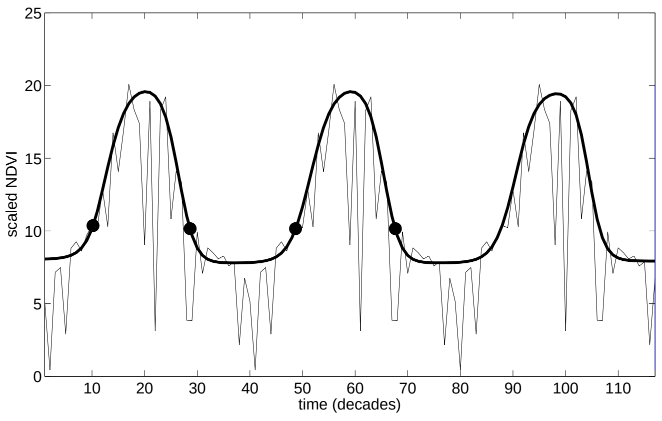


Figure 14: *To extract seasonality parameters from one year of data the time-series has been duplicated to span three years.*

4.3 Defining start and end of season

In Timesat 3.3 four methods are used for determining when the seasons start and end. (1) The first method is based on the seasonal amplitude, defined between the base level and the maximum value for each individual season. The start occurs when the left part of the fitted curve has reached a specified fraction of the amplitude, counted from the base level. The end of season is defined similarly, but for the right side of the curve. (2) In the second method, the start/end of season occurs when the curve has reached an absolute value, defined in the units of the data. (3) The third method is based on the relative amplitude for the whole time series. This amplitude is calculated as the difference between the robust mean maximum and the robust mean base level (the means of values when excluding the 10 % lowest and highest values). The start/end occur when the curve has reached a specified fraction of this relative amplitude. In contrast to method 1, this method will generate start/end vegetation index values that are identical for all seasons of a point (pixel), however, the value can vary between different points (pixels). (4) In the fourth method the start/end occur when the curve crosses the STL trend line. The methods are illustrated in Figure 15. The choice of method and threshold values depends on the vegetation index used, the biological threshold chosen (e.g. start of season defined as leaf area index exceeding a certain value), and how this threshold translates into a vegetation index value or fraction of amplitude for the studied vegetation type.

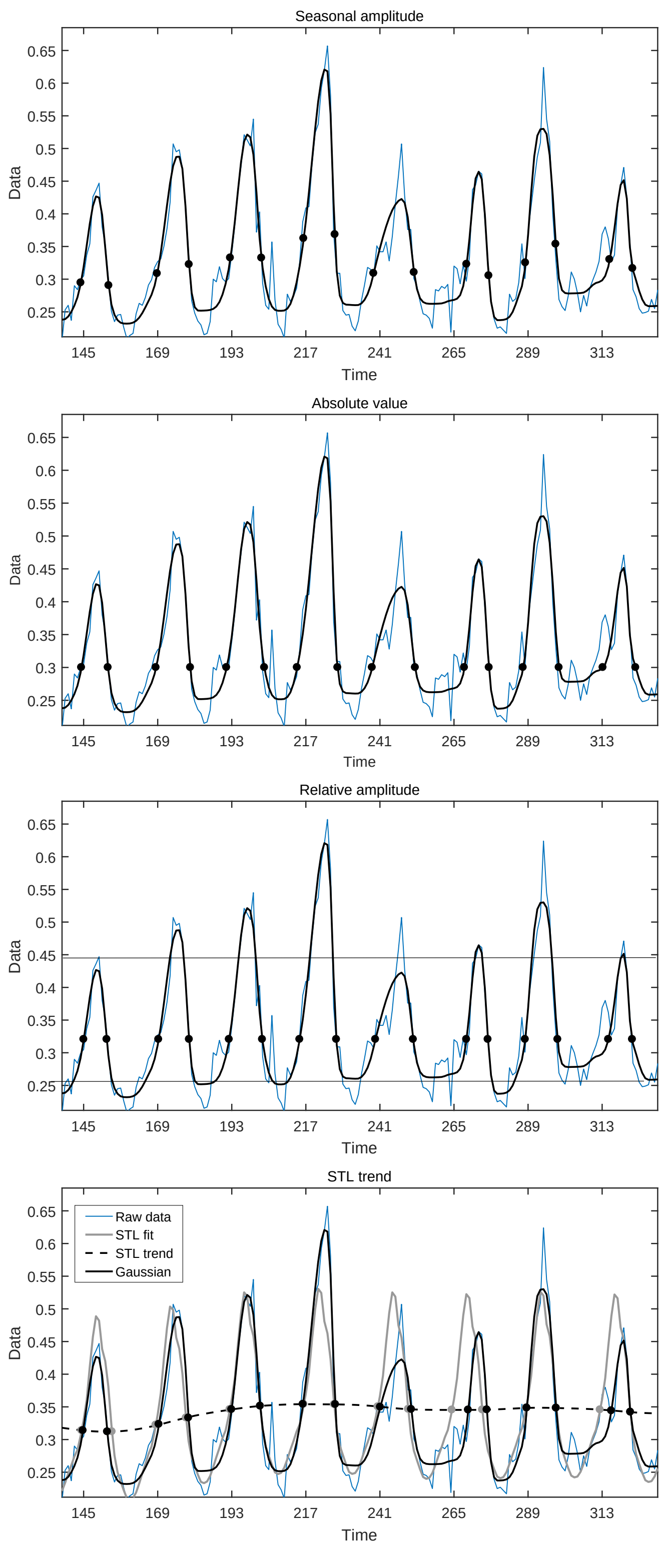


Figure 15: *Four methods for determining beginning and end of seasons. From the top: 1) Seasonal amplitude = 0.3. 2) Absolute value = 0. 3) Relative amplitude = 0.3; horizontal lines denote robust mean maximum and base level. 4) Crossing of the STL trend curve (STL fits drawn).*

4.4 Extracted seasonality parameters

In the current version of TIMESAT a number of key seasonality parameters such as the time of the start and end of the season, the largest value, and the amplitude are computed for each of the full seasons in the time-series. Some of these parameters are displayed in Figure 16. The use of the fitted function gives more stable measures, where effects of noise have been reduced. To rule out errors a number of checks on consistency of the parameters are done. **Please note, once again, that a time-series spanning n years will give seasonality parameters for the $n-1$ center-most seasons.**

Below are definitions of all the extracted seasonality parameters. There are of course no unique definitions of seasonality parameters and different researchers may argue for different ways of extracting and validating these parameters. However, the importance of these parameters lies in the possibility to map out spatial or temporal changes in the vegetation cover resulting from climatic or land use changes.

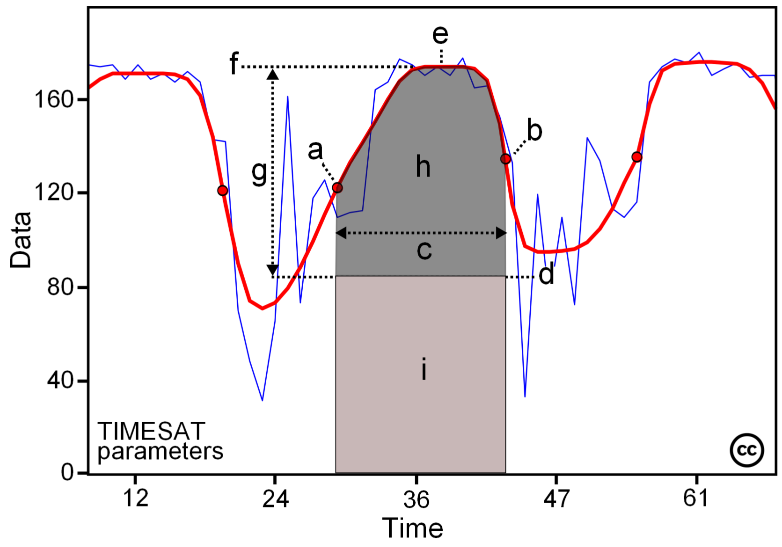


Figure 16: *Figure 1. Some of the seasonality parameters generated in TIMESAT: (a) beginning of season, (b) end of season, (c) length of season, (d) base value, (e) time of middle of season, (f) maximum value, (g) amplitude, (h) small integrated value, (h+i) large integrated value. This figure is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 2.5 Sweden License. It is free to copy and use in other work.*

1. **time for the start of the season;** time for which the left edge has increased to a user defined level (often a certain fraction of the seasonal amplitude) measured from the left minimum level.

2. **time for the end of the season;** time for which the right edge has decreased to a user defined level measured from the right minimum level.

3. **length of the season;** time from the start to the end of the season.

4. **base level;** given as the average of the left and right minimum values.

5. **time for the mid of the season;** computed as the mean value of the times for which, respectively, the left edge has increased to the 80 % level and the right edge has decreased to the 80 % level.

6. **largest data value for the fitted function during the season;** may occur at a different time compared with 5.

7. **seasonal amplitude;** difference between the maximum value and the base level.

8. **rate of increase at the beginning of the season;** calculated as the ratio of the difference between the left 20 % and 80 % levels and the corresponding time difference.

9. **rate of decrease at the end of the season;** calculated as the absolute value of the ratio of the difference between the right 20 % and 80 % levels and the corresponding time difference. The rate of decrease is thus given as a positive quantity.

10. **large seasonal integral;** integral of the function describing the season from the season start to the season end. Note that the large integral has no meaning when part of the fitted function is negative.

11. **small seasonal integral;** integral of the difference between the function describing the season and the base level from season start to season end.

12. **value for the start of the season;** value of the function at the time of the start of the season.

13. **value for the end of the season;** value of the function at the time of the end of the season.

5 Aspects of processing

There are many aspects of data processing for achieving optimal results. One aspect is to choose the best method, Savitzky-Golay, asymmetric Gaussian or double logistic, for a given data set. Another aspect is to fine tune the program settings guiding the processing. Below we will in broad terms discuss the characteristics of the different methods implemented in Timesat. After that we will go on to discuss the settings in detail.

5.1 Characteristics of the processing methods

In Timesat the user can choose between adaptive Savitzky-Golay filtering and fits to asymmetric Gaussians and double logistic functions. In contrast to functions resulting from Fourier methods, the resulting functions in Timesat are local in the sense that they are able to capture inter-annual changes, i.e. changes in seasonal timing between years. This property makes them suitable for studying vegetation dynamics.

To reduce the influence of clouds and atmospheric constituents satellite derived time-series are often maximum-value composites (MVC), where the largest value in a defined period, e.g. 10-day period (decade), is selected to represent the period. The time-series are thus, in a strict meaning, not evenly sampled. Most processing methods ignore the effects of this uneven sampling. The methods implemented in Timesat are all based on least-squares fits and it is, in principle, possible to process time-series that are unevenly sampled using the time stamp associated with the MVC. This feature is however not implemented in the current version of Timesat. Instead all time-series are treated as if they were sampled at an even rate.

The different processing methods in Timesat have different strengths and weaknesses. For comparatively smooth time-series the three different processing methods often give very similar results, and which one to use should be carefully tested using the graphical interface in Timesat (see section 9.4). If the time-series is smooth, but with a plateau indicating that the underlying signal is composed of two vegetation signals, then the more local Savitzky-Golay filter is the preferred method (Jönsson and Eklundh 2003). For noisy time-series the Savitzky-Golay method sometimes yields undesirable results. In these cases fits to the asymmetric Gaussians or double logistic functions may be the better choice. The final choice of methods depends on the character of the input data and has to be decided by inspecting how well the fitted functions match the original data.

The performance of different processing methods have been evaluated in a study by Hird and McDermid (2009). The conclusion is that the methods in Timesat are highly competitive and that they to a great extent preserve the signal integrity.

5.2 Controlling the processing: input settings

The processing in Timesat is controlled by a number of settings. Depending on these settings, such as degree of adaptation to the upper envelope or, in the case of Savitzky-Golay filtering, the size of the moving window, the results may be very different. Here we list the input settings. Settings are read from a settings file (`*.set`), that can be edited by hand or manipulated by the program `TSM_settings` (see section 9.5). The settings can also be generated with the Timesat graphical user interface (GUI). The GUI allows the user to actively experiment with the settings to find the optimal combination for the time-series at hand. The settings in the GUI are then transferred to the settings file (`*.set`). The GUI and how to use it to process different types of data is discussed at length in section 9.4 in Part III of this document. A detailed description of each setting is given in the next section. Please note that the settings file format is new in version 3.3.

Row	Example	Short description	Explanation
1	`Version: 3.3`		Keeps track of the settings file version
2	`west_africa`	Job name	Job name (no blanks) - max 100 chars. This will determine the name of output files from TIMESAT
3	1	Image/series mode (1/0)	1 = image mode, 0 = ASCII time-series
4	0	Trend (1/0)	1 = STL trend fitting activated. Overrules choice of fitting method (row 32).
5	1	Use quality data (1/0)	1 = use quality data, 0 = do not use quality data
6	`datalist.txt`	Data file list/name	Name, followed by `%`, of file list (for images) or data file name (for ASCII data).
7	`quallist.txt`	Quality file list/name	Name, followed by `%`, of quality list (for images) or quality file name (for ASCII data)
8	1	Image file type	1 = 8-bit unsigned integer, 2 = 16-bit signed integer, 3 = 32-bit real
9	0	Byte order (1/0)	0 = little endian byte order, 1 = big endian byte order (for 16-bit integers)
10	200 200	Image dimension	No. of rows in image, and no. of columns per row
11	111 120 91 100	Processing window	Window to process (start row, end row, start column, end column)
12	3 36	Years and points per year	No. of years and no. of points per year
13	1 255	Valid data range	Lower and upper data values for valid range. Data outside range will be assigned weight 0
14	1 12 0.1	Quality range 1 and weight	Lower and upper values for quality class 1 and assigned weight
15	13 22 0.5	Quality range 2 and weight	Lower and upper values for quality class 2 and assigned weight
16	23 31 1	Quality range 3 and weight	Lower and upper values for quality class 3 and assigned weight
17	0	Amplitude cutoff value	Cutoff for low amplitude. Series with amplitude smaller than this value will not be processed. 0 processes all data
18	0	Debug (3/2/1/0)	Debug flag. 1 - 3 = print debug data, 0 = do not print debug data
19	1 1 0	Output files (1/0 1/0 1/0)	Flags for output data (seasonality, fitted data, and original data)
20	0	Use land cover (1/0)	1 = use land cover map, 0 = do not use land cover map
21	landcoverdata	Name of land cover file	Name, followed by `%`, of land cover file

22	1	Spike method (3/2/1/0)	Spike method. 3 = weights from STL multiplied with original weights, 2 = weights from STL, 1 = method based on median filtering, 0 = no spike filtering
23	2	Spike value	If spike method = 1 the spike value determines the degree of spike removal. A low value will remove more spikes
24	0	STL stiffness value	Parameter for STL trend stiffness. Varies between 1.0 and 10.0; default = 3.0.
25	2	No. of land cover classes	No. of land cover classes (if land cover data are used)
26	************	Separator	After separator comes class specific parameters
27	1	Land cover code for class 1	Land cover code for class 1
28	1	Seasonality parameter (0-1)	A value near 1 will attempt to fit one season per year, a value close to zero will attempt to fit two seasons
29	3	No. of envelope iterations (3/2/1)	No. of iterations for upper envelope adaptation (3,2,1). Choosing 1 means no envelope adaptation.
30	2	Adaptation strength (1-10)	Strength of the envelope adaptation. 10 is the maximum strength
31	0 0	Force to minimum (1/0) and value of minimum	Force to minimum. 1 = points below given minimum value will be forced to the specified minimum value. 0 = no forcing to value
32	3	Fitting method (3/2/1)	Fitting method. 3 = double logistic, 2 = asymmetric Gauss, 1 = Savitzky-Golay If STL trend fitting activated (row 4) this overrides the fitting method.
33	1	Weight update method	Weight update method (not in use)
34	4	Window for Savitzky-Golay	Half window for Savitzky-Golay filtering. A large value of the window will give a high degree of smoothing
35	0	Reserved	Not in use
36	0	Reserved	Not in use

37	1	Season start start/end method (4/3/2/1)	Method for determining start/end of season based on intersection of the fitted curve. 4 = STL trend: at the intersection with the trend line from STL. 3 = Relative amplitude: at the point where the curve intersects a proportion of the relative seasonal amplitude. 2 = Absolute value: at the point where the curve intersects an absolute value in units of the data. 1 = at the point where the curve intersects a proportion of the seasonal amplitude.
38	0.5 0.5	Season start / end values	Values for determining season start/end If start method is 1 or 3 the values must be between 0 and 1
39 – 51		Separator and data for class 2	Same as rows 26–38, but for class 2
52 – 64		Separator and data for class 3	Same as rows 26–38, but for class 3
· · ·		· · ·	· · · etc. for a maximum of 255 classes

5.3 Description of input settings

For convenience the settings file has the same input entries independent of whether we process sequential data in an ASCII file or data in image files. If, for example, we process sequential data in an ASCII file not all entries are actually used. For entries that are not needed dummy values in the correct format should be given. Having the same input entries make it easier to edit the input file. It is possible to add comments at the end of each row, indicating the meaning of each setting. An example of a settings file can be found in the sample data (see section 7.2).

Row 1, Version: 3.3

Row 2, Job name
Character string that will be used to label output files from TIMESAT.

Row 3, Image/series mode (1/0)
1 = image mode, 0 = ASCII file with time-series.

Row 4, Trend (1/0)
1 = STL trend fitting activated. Overrules choice of fitting method (row 32).

Row 5, Use quality data (1/0)
1 = use quality data, 0 = do not use quality data. As described in section 3.2 quality data consists of numbers that can be translated into weights.

Row 6, Data file list/name
Running in image mode the user should prepare a file that on the first row gives the total number N of vegetation index images and then the path and name of each of the N images (compare figure 3). The structure of the file is shown below

```
N
path\imagename_1
path\imagename_2
  ....
path\imagename_N
```

The name, followed by `%`, of the so prepared file should be supplied on row 6. Running sequential data the user should instead give the name of the ASCII file containing this data. The structure of the ASCII file is specified in section 2.1.

Row 7, Quality file list/name
Relevant if quality data are used (specifications on rows 14–16). Running in image mode the user should prepare a file that on the first row gives the total number N of quality images and then the paths and names of the quality images. The file has the same structure as the file listing vegetation index images. The name, followed by `%`, of the prepared file should be supplied on row 7. Running sequential data the user should specify the ASCII file containing the quality data. If quality data are not used the user should simply input any dummy name.

Row 8, Image file type
Relevant if in image mode. Please specify the data types of the images where 1 = 8-bit unsigned integer, 2 = 16-bit signed integer, 3 = 32-bit real (see also section 10.15). If not in image mode the user may simply input the value 0.

Row 9, Byte order (1/0)
Relevant if in image mode. Please specify the byte order where 0 = little endian byte order, 1 = big endian byte order (for 16-bit signed integers). If not in image mode the user may simply input the value 0.

Row 10, Image dimension
Relevant only in image mode. In this case the first number gives the number of rows in the images and the second number the number of columns. If not in image mode the user may simply input the numbers 0 and 0.

Row 11, Processing window
Relevant only in image mode. Four integers should be supplied giving start row, end row, start column, and end column for the area to be processed. If not in image mode the user may simply input four zeros.

Row 12, Years and points per year
Input the number of years a time-series is spanning and also the number of points per year. There are some checks to see that numbers are consistent with other data. For example the product of the years and the number of points per year should be equal to the number N of supplied images on rows 6 and 7.

Row 13, Valid data range
Specify lower and upper data range. Data outside the specified range will be assigned weight 0. By choosing these values carefully one may for example avoid that water pixels are processed.

Rows 14 – 16, Quality range and weight
Relevant if quality data are used. Remotely sensed data often come with some cloud classification or other quality indicator representing broad quality classes (see section 3.2). The quality indicators in each class are transformed into weights, determining the importance of the associated data values in the least-squares fits. In row 14 the user should supply lower and upper values for quality class 1 and the assigned weight. In row 15 the user should supply lower and upper values for quality class 2 and the assigned weight Finally, in row 16 lower and upper values for quality class 2 and the assigned weight should be given. If quality data are not used one may input three zeros in each of rows 14, 15, and 16. Three quality classes are used in TIMESAT.

Row 17, Amplitude cutoff value
Cutoff for amplitude. Time-series with smaller average amplitude than the cutoff will not be processed. Useful for excluding areas with minimal seasonal variation, e.g. deserts. Set to 0 to process all data.

Row 18, Debug (0-3)
Debug flag. 0 = do not print debug data (recommended); 1 = print certain debug parameters to the screen; 2 = print certain debug parameters to file `debug2_jobname`; 3 = if a crash occurs the position of the problematic time-series as well as the time-series itself is written to `debug3_jobname`.

Row 19, Output files (1/0 1/0 1/0)
Flags for output data. The first flag determines if seasonality data should be printed or not, the second flag determines if fitted functions should be printed, and finally the third flag

determines if the original time-series should be printed or not.

Row 20, Use land cover (1/0)
1 = use land cover map, 0 = do not use land cover map. Relevant only in image mode. Processing an ASCII file the user may simply supply the value 0. More information is given in section 9.1.

Row 21, Name of land cover file
Name and path of the land cover file. If a land cover map is not used the user may supply a dummy name. The name should be followed by `%`. The landcover file must have the same format and dimensions as the input image files.

Row 22, Spike method (3/2/1/0)
There are three different methods implemented to detect spikes in data and set corresponding weights to zero. 3 = weights from STL-decomposition (the full time-series is divided into a seasonal- and a trend component, data values that do not fit this pattern are assigned low weights, see Cleveland *et al.* 1990 for detailed information) multiplied with original weights, 2 = weights from STL-decomposition. 1 = method based on median filtering as described in section 3.3. 0 = no spike detection.

Row 23, Spike value
Relevant only if spike method 1 is used. Data values that differ from the median value by more than the spike value multiplied with the standard deviation of y and that are different from the left and right neighbors are removed (assigned weight 0). A normal setting of the spike value is 2. A lower spike value will remove more data values, some of which may be correct. If spike method 1 is not used the user can supply any value.

Row 24, STL stiffness value
This value regulates the stiffness of the STL trend variable. The default is 3.0. A smaller value decreases stiffness, and a larger value increases stiffness.

Row 25, No. of land cover classes
Number of land cover classes. Relevant only if a land cover map is used. If a land cover map is not used the user may put 1 in this entry.

Row 26, Separator
This row contains a separator. On the rows following the separator parameters are given that are specific to each land cover class. If no land cover map is used all time-series will be treated as belonging to the first class.

Row 27, Land cover code for class 1
Land cover code for class 1. Time-series for all pixels in the image with this land cover code will be processed with the parameter settings in rows 28–38. If there is no land cover file or if processing sequential data in an ASCII file all time-series will be processed with the parameter settings in rows 28–38, i.e. as if they belonged to land cover class 1.

Row 28, Seasonality parameter
This parameter guides how the secondary maximum in the determination of the number of seasons is treated (see section 3.5). A value 1 of the parameter will force the program to treat all data as if there is one season per year. A small value of the parameter will attempt to fit two seasons a year. If there are images covering areas with both one and two vegetation

seasons, as may be the case for images on continental scale, it is advisable to separate these areas in two different land cover classes using a high value of the seasonality parameter for the class with one vegetation season and a low value for the class with two vegetation seasons.

Row 29, No. of envelope iterations
The function fits can be made to approach the upper envelope of the time-series in an iterative procedure (see section 3.4). Specifying 1 for the number of envelope fits there is only one fit to data and no adaptation to the envelope. Specifying 2 or 3 there are, respectively, one and two additional fits where the weights of the values below the fitted curve is decreased forcing the fitted function toward the upper envelope.

Row 30, Adaptation strength
The adaptation strength is a number between 1 and 10 indicating the strength of the upper envelope adaptation. 10 gives the strongest adaptation to the upper envelope and 1 gives no adaptation. Strong adaptation, especially combined with 3 envelope iterations, may put too much emphasis on single high data values leading to bad results. The adaptation strength needs to be fine tuned for given data, but a normal adaptation value is around 2 and 3.

Row 31, Force to minimum (1/0) and value of minimum
At northern or southern latitudes time-series may during the dark season be affected by high sun zenith angles and/or pertinent clouds, giving unphysically low values during long periods of time. In these cases it may sometimes be useful to force the fitted function to a user specified minimum (or off-season) value. This is done by giving 1 for the first entry followed by the minimum value. If the user specifies 0 for the first entry there will be no forcing to the minimum value.

Row 32, Fitting method (3/2/1)
Indicate fitting method. 3 = double logistic function, 2 = asymmetric Gaussian, 1 = Savitzky-Golay filtering. Which method to use is determined by the properties of the time-series (compare discussion in section 5.1). Different methods can be used for different land cover classes. If STL trend fitting is activated (row 4), this overrides the fitting method setting.

Row 33, Weight update method
Weight update method; not in use. The user may simply input 1.

Row 34, Window size for Savitzky-Golay
If Savitzky-Golay filtering is used (see section 3.6) the half-window n needs to be set. This integer value should be seen in relation to the total number data values during the year. A rough guide value is around `floor`($nptperyear/4$). A large value of the window gives a high degree of smoothing, but affects the possibility to follow a rapid change in data in the beginning of the growth season.

Row 35, Reserved
Reserved for future use. The user should supply the value 0.

Row 36, Reserved
Reserved for future use. The user should supply the value 0.

Row 37, Season start/end method (4/3/2/1)
Method for defining the start/end of seasons (see further explanations in section 4.3). 4 = STL trend, 3 = relative amplitude, 2=absolute value, 1=seasonal amplitude. For methods 3,

2 and 1 the threshold values for start and end respectively are specified on row 38.

Row 38, Season start/end values
For start / end methods 3 and 1 please supply the threshold values as a proportion of amplitude, ranging between 0 and 1. For method 2 specify absolute values in data units. Not used for method 4 (supply any values).

Rows 39 – 51, Data for class 2
Same information as on rows 26 – 38 (including the separator) etc.

6 Output data

Depending on the input parameter settings Timesat outputs files containing: original (raw) time-series read from the ASCII files or extracted from the images, time-series from fitted functions, determined seasonality parameters, and debug information. We start to discuss the output files resulting from processing image data (see section 2.2). Output files obtained by processing ASCII files (see section 2.1) can be seen as a special case and are treated at the end of this section.

6.1 Files with time-series: *.tts

The file with the original (raw) time-series copied from the images has the name `jobname_raw.tts`. In a similar way the file with the time-series constructed from the fitted functions has the name `jobname_fit.tts`, where `jobname` is the name given in the beginning of a run. Both files are binary and data are organized according to Figure 17. In the files *nyears, nptperyear, rowstart, rowstop, colstart, colstop* are integers specifying, respectively, the number of years spanned by the time-series, the number of data values in one year, and the area in the image that has been processed. Finally, *row, col* specifies the position of the time-series in the image. The above integers are written in the format `int32`. The time-series $y_1, y_2, \ldots, y_N$ for each of the pixels (row, col) in the area are written in single precision `real*4`. Data are given by row meaning that the column index varies faster than does the row index.

$$\begin{array}{l}
nyears \;\; nptperyear \;\; rowstart \;\; rowstop \;\; colstart \;\; colstop \\
row_1 \;\; col_1 \\
y_1 \;\; y_2 \;\; y_3 \;\; \cdots \;\; y_{N-1} \;\; y_N \\
row_2 \;\; col_2 \\
y_1 \;\; y_2 \;\; y_3 \;\; \cdots \;\; y_{N-1} \;\; y_N \\
\quad\;\; \vdots \\
row_M \;\; col_M \\
y_1 \;\; y_2 \;\; y_3 \;\; \cdots \;\; y_{N-1} \;\; y_N
\end{array}$$

Figure 17: *Data structure of binary files* (`*.tts`) *containing raw time-series and time-series from fitted functions. The first line of the file gives information about the number of years spanned by the time-series, nyear, the number of data values per year, nptperyear, and the*

spatial extension of the area.

The time-series files can be read by the program `TSM_viewfits` (see sections 9.8 and 10.13), and it is possible to step forward in the file and check the TIMESAT fits pixel by pixel.

6.2 Files with seasonality parameters: *.tpa

The file with the extracted seasonality parameters has the name `jobname_TS.tpa`. The data structure of the file is displayed in Figure 18. The integers *nyears, nptperyear, rowstart, rowstop, colstart, colstop, row, col* have the same meaning as above. The integer n gives the number of full seasons for which seasonality information has been determined. The seasonality parameters $p_1, p_2, \ldots, p_{13}$ (cf. section 4.4) are written in single precision `real*4`.

$$
\begin{array}{l}
nyears \quad nptperyear \quad rowstart \quad rowstop \quad colstart \quad colstop \\
row_1 \quad col_1 \quad n_1 \\
\left.\begin{array}{ccccccccccccc}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
 & & & & & \vdots & & & & & & & \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13}
\end{array}\right\} \begin{array}{l} n_1 \text{ lines with values} \\ \quad \text{one line per season} \end{array} \\
row_2 \quad col_2 \quad n_2 \\
\left.\begin{array}{ccccccccccccc}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
 & & & & & \vdots & & & & & & & \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13}
\end{array}\right\} \begin{array}{l} n_2 \text{ lines with values} \\ \quad \text{one line per season} \end{array} \\
\qquad \vdots \\
row_M \quad col_M \quad n_M \\
\left.\begin{array}{ccccccccccccc}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
 & & & & & \vdots & & & & & & & \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13}
\end{array}\right\} \begin{array}{l} n_M \text{ lines with values} \\ \quad \text{one line per season} \end{array}
\end{array}
$$

Figure 18: *Data structure of binary files* `jobname_TS.tpa` *containing seasonality parameters extracted from the fitted functions. The first line of the file gives information about the number of years spanned by the time-series,* nyear, *the number of data values per year,* nptperyear, *and the spatial extent of the area. For each pixel* (row, col) *in the area seasonality parameters are given for the specified number of seasons.*

The data are given by row meaning that the column index varies faster than does the row index: first comes data for all columns belonging to the first row, then comes data for all columns for the second row etc.

6.3 Files with output from STL trend analysis

Two types of files with fitted data from the STL smoothing are generated when the option for trend analysis is chosen: `jobname_STLfit.tts`, containing the seasonal component of the STL fit, and `jobname_STLtrend.tts`, containing the trend line of the STL fit. In addition the file `jobname_STL.tpa` is generated, containing seasonality parameters for the seasonal component. The formatting of these files are identical to the `.tts` and `.tpa` files described in 6.1 and 6.2.

6.4 Extracting images of seasonality parameters

The timings of seasons do not always follow the calender year. For example a vegetation season may start in October, peak in December, and fall off in March the following year. To generate images of seasonality parameters e.g. vegetation amplitude for this season from the file with seasonality parameters, a time window containing the season must be defined (see Figure 19). One searches the seasonality file by looping over the pixels. For each pixel there is then a loop over all the seasons. The season falling in the time window is the desired one and the seasonality parameter is extracted for this season and written to the image file. The algorithm for generating an image file can be described as follows:

1. Give spatial extension of the area
2. Give the time window for the season
3. Specify which seasonality parameter should be displayed
4. Loop over pixels (row, col) in the defined area
5. For each pixel loop over the seasons
   (a) Read seasonality parameters for the season
   (b) If the peak value of the season is within the specified time window, write the value of the specified seasonality parameter to the image file. Note that from version 3.3 the peak value of the season is used rather than the start and end of the season. This reduces the risk of missing the season entirely.
6. Display image file

The user is advised to make the time window large enough to allow for a certain variation in the start and end of the season over the processed area. The extraction of images is done using the program `TSF_seas2img` (see section 9.8).

6.5 Output files from ASCII data

The format and structure of output files from runs with time-series given in ASCII files (see section 2.1) are the same as the format and structure of output files from runs where the time-series are extracted from a sequence of one-column images. Thus for the output files *col* is always 1 and *row* specifies the sequential number of the time-series in the ASCII file. Also, for output files from runs with time-series given in ASCII files $rowstart = 1$, $rowstop = nts$, $colstart = colstop = 1$, where nts is the number of time-series in the ASCII file.

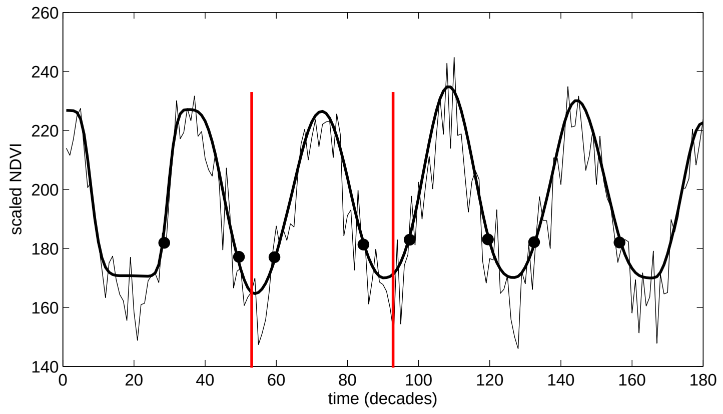


Figure 19: *To extract information from the seasonality file for a specific season a time window must be defined. The season for which the maximum of the season lies within the window is the desired one.*

6.6 Index files: *.ndx

TIMESAT generates index files that allow for faster access of the output data files. The output files can be very large, and the index files can considerably speed up access to specific locations. This is particularly noticed when plotting data using e.g. the routines `TSM_printseasons` and `TSM_viewfits`. The index files have the extension `.ndx`, and have the following formats: For files with time series (`*.tts`):

$$\begin{array}{lll} row_1 & col_1 & loc_1 \\ row_2 & col_2 & loc_2 \\ \\ row_n & col_n & loc_n \end{array}$$

where *row* and *col* are the pixel locations containing data in the `.tts` file, and *loc* is the location of the beginning of the row/col numbers in bytes from the beginning of the `.tts` file. row, col and loc are double integers (`int64`).

For files with seasonality data (`*.tpa`):

$$\begin{array}{llll} row_1 & col_1 & nseas_1 & loc_1 \\ row_2 & col_2 & nseas_2 & loc_2 \\ \\ row_n & col_n & nseas_n & loc_n \end{array}$$

where *row*, *col* are pixel numbers, *nseas* is the number of seasons for this pixel, and *loc* is the location of the beginning of the row/col numbers in bytes from the beginning of the `.tpa` file. $row, col, nseas$ and loc are double integers (`int64`).

Part III
Software User's Guide

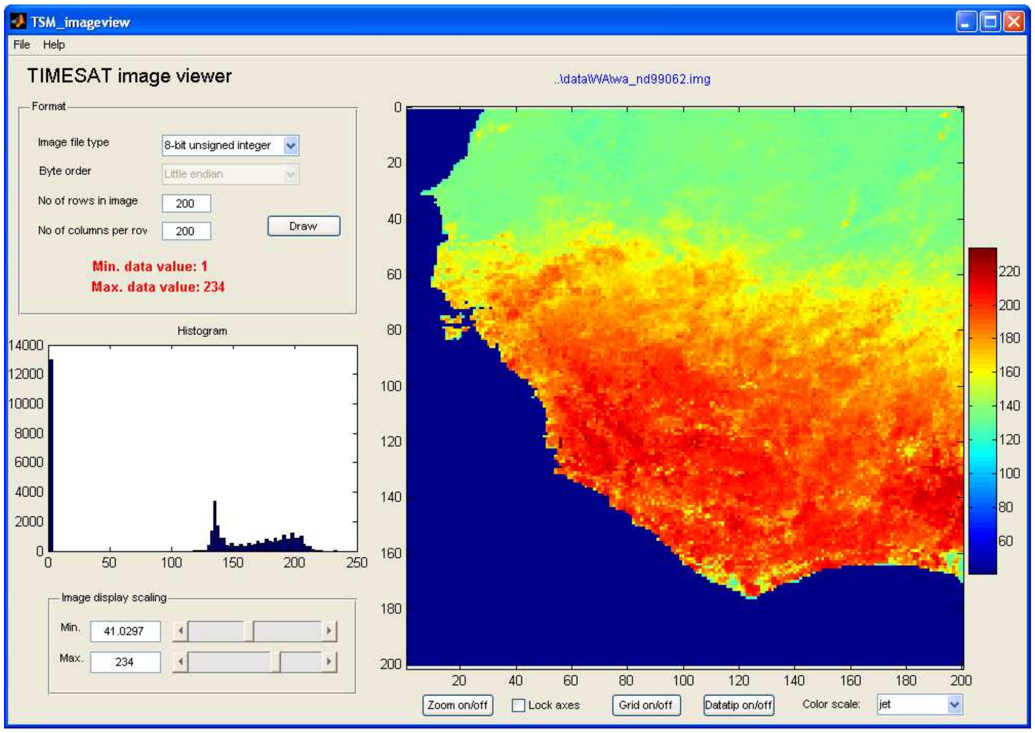


Image file type 8-bit unsigned integer

Byte order Little endian

No of rows in image 200

No of columns per rov 200

Min. data value: 1

Max. data value: 234

Min. 41.0297

Max. 234

Color scale: jet

7 Installation of TIMESAT and program structure

7.1 System requirements

The TIMESAT package consists of routines developed in Matlab and Fortran. It has been developed under Windows and tested also under Linux. Graphically, there are small differences between the two operating systems, but functionally not. The programs allocate memory dynamically, and very large data sets can be processed. An exception is the module `TSM_GUI` that loads all requested data at once and thus may experience memory limitations. To serve users that do not have Matlab, we are supplying a compiled version of the Matlab routines that run directly under Windows and Linux. The Fortran programs are also pre-compiled, and executables are available both for Windows and Linux.

7.2 Installation

To install TIMESAT create a directory on the hard drive, as a suggestion with the name `timesat`, and unpack the zipped TIMESAT file in the created directory. The zipped file can be downloaded from the TIMESAT homepage. After unpacking the zipped file the following folders will be present:

Top level	**Sub-level**	**Contents**
`timesat33`	`\compiled`	Executables for pre-compiled version of TIMESAT under Win64 and Linux 64 for users without Matlab
	`\data`	Sample data
	`\data\single`	Sample data stored in single ASCII files
	`\data\wa`	Sample images for West Africa
	`\documentation`	Documentation and reprints of articles
	`\run`	Optional startup directory for running TIMESAT (empty on installation)
	`\timesat_fortran\main`	Fortran executables for main processing routines
	`\timesat_fortran\tools`	Fortran executables for post-processing routine
	`\timesat_matlab\main`	Matlab executables for main processing routines
	`\timesat_matlab\tools`	Matlab executables for some GUI routines

Installation for users with Matlab

After starting Matlab, the TIMESAT Matlab directories need to be appended to the *Matlab Path* (use *Add with Subfolders* in *Set Path* and select `timesat_matlab` to include both the subfolders `timesat_matlab\main` and `timesat_matlab\tools`. TIMESAT, including the Fortran routines, can then be run from any working directory in Matlab. An example is given in section 9.6.

If you wish to run any of the Fortran programs standalone (outside the Matlab working environment) you can copy the executables (`*.exe` on Windows and `*.x64` on Linux) from `timesat_fortran\main` and `timesat_fortran\tools` to your working directory, open a command window from the working directory, and execute the programs from there by typing their names. Alternatively you must open a command window prompt and append the Fortran directories to the path before executing them. An example of how to append the DOS path in the latter case is given in section 9.6.

Installation of pre-compiled version for users without Matlab

We are supplying executable files of TIMESAT that can run directly under Windows or Linux without having Matlab installed. These files are contained in the `\compiled\Win64` subfolder for Windows users and `/compiled/Linux64` subfolder for Linux users. It is first necessary to install a runtime engine called the Matlab Compiler Runtime (MCR). Note that the included version of MCR must matche the compiler version used by us for creating TIMESAT and that other versions of MCR may not work. Also note that you will need administrative privileges on the machine for installing MCR. We do not recommend installing MCR on machines that already have Matlab installed. Please go to the Matlab Compiler technical reference pages for more information about MCR. Installation and startup differs between Windows and Linux:

Windows users: Double click on file `MCRinstaller.exe` found under `\compiled\Win64`. You might receive the following message *".NET Framework is not installed. If you require it, select Cancel and install .NET framework first. Otherwise select OK to continue"*. You can just continue the installation by pressing *OK*. To start TIMESAT double click on the file `TIMESAT.exe` in the `\compiled\Win64` folder. You will be asked to supply the name of the installation folder. Give the name of the folder where TIMESAT was installed, e.g. `C:\timesat33`. In the TIMESAT menu system use *File, Preferences* to navigate to your working directory.

Linux users: Execute the file `MCRInstaller.bin` found under `/compiled/Linux64`. This will install MCR into a folder. Suppose, for simplicity, that the MCR installation folder is `/usr/local/mcr_folder`. Invoke, in the `/compiled/Linux64` folder, the TIMESAT shell script followed by the MCR installation folder

```
./run_TIMESAT.sh  /usr/local/mcr_folder
```

The shell script sets up the environment and executes TIMESAT. During the process you will be asked to supply the name of the TIMESAT installation folder e.g. `/home/username/timesat33`. In the TIMESAT menu system use *File, Preferences* to navigate to your working directory.

More information about the Linux installation is given in the file `readme.txt` in the subfolder `/compiled/Linux64`.

8 Program and processing overview

8.1 Processing logic

TIMESAT consists of several different routines written in Matlab and Fortran. Graphics-oriented programs are coded in Matlab, and programs for processing large data sets are coded in Fortran to achieve fastest possible execution. The general logic of the processing is given in Figure 20 and is briefly explained here. The main processing steps in Figure 18 are described by numbers in the left margin of the figure:

1. Preparation of input data and previewing of binary images. Two types of data are accepted, ASCII files of single or multiple data series, or full sets of binary images in time sequence. To view image data the program `TSM_imageview` is used.

2. Running Timesat for selected image pixels or time-series using the program `TSM_GUI`. This allows the user to check the quality of the input data and to select suitable fitting algorithms and settings for running the full data sets.

3. Creation of an ASCII settings file. This is done with the program `TSM_settings`, accessible from `TSM_GUI` or as a stand-alone program. The settings control how Timesat treats the input data. Two types of input settings exist: general settings that control the processing of all pixels, and class-specific settings that control processing for a given land cover class (cf. sections 5.2 and 5.3).

4. Running Timesat for full data sets using the settings file generated in the previous step. This is normally done from a command window using the Fortran program `TSF_process`. Output data consist of binary files for seasonality data and fitted data, each file containing the result for all input series e.g. all image pixels (cf. sections 6.1 and 6.2).

5. Generation of output images from the Timesat output files. This includes viewing of seasonality data or fitted data for single pixels and creation of image files of seasonality and fitted data for given time periods. Several routines are available for these purposes (cf. section 6.4). For a full description of the processing steps please go to the appropriate sections in chapter 10.

8.2 Naming convention of programs

Programs coded in Matlab are all given the prefix `TSM_` (Timesat Matlab). Programs coded in Fortran are all given the prefix `TSF_` (Timesat Fortran).

8.3 Program versions

Timesat 3.3 was written in Matlab ver. 2016b. Fortran programs were compiled with the Intel Fortan Compiler XE 12.1 (Windows) and 12.1 (Linux) compiler.

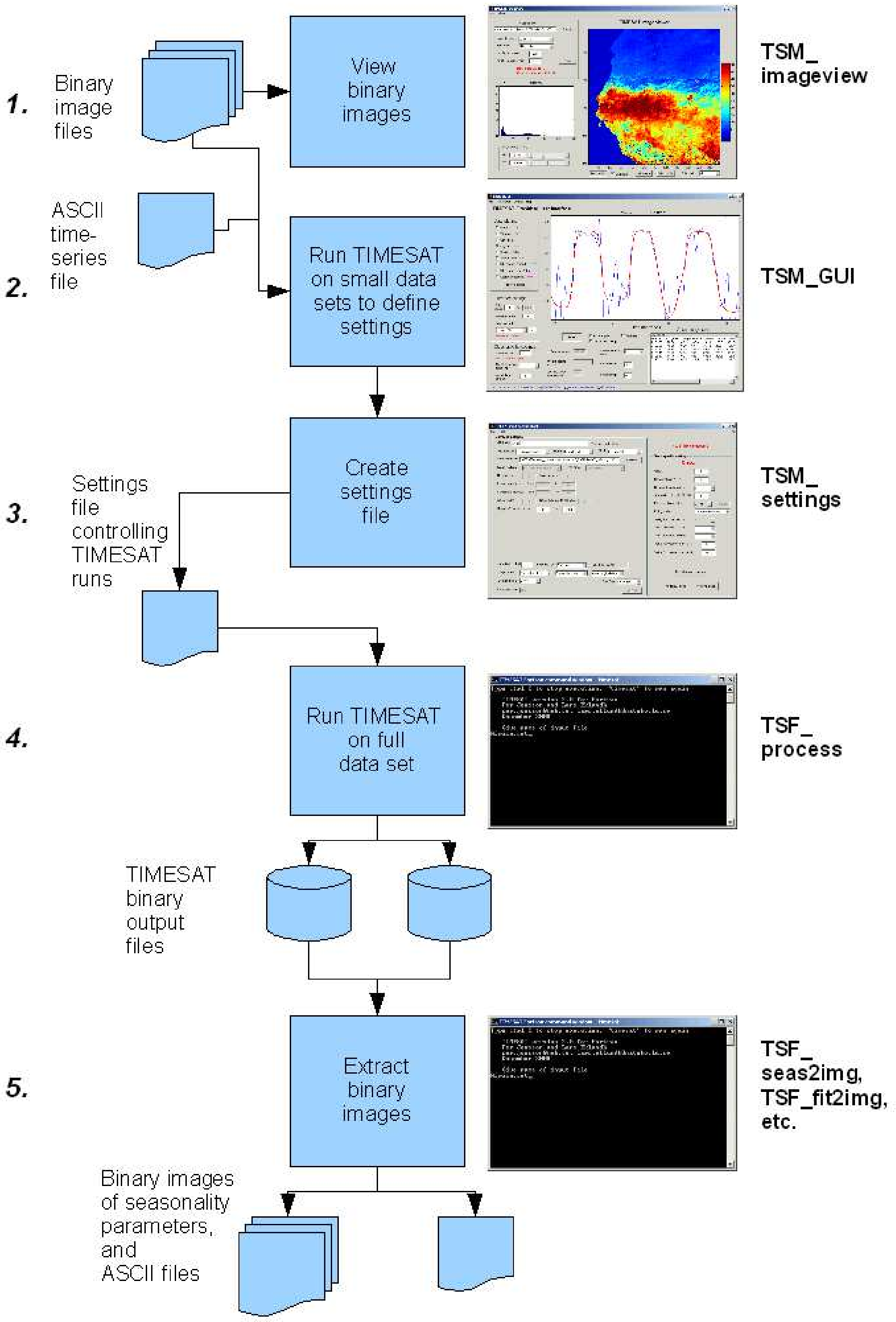

1.
Binary image files
View binary images
TSM_ imageview
ASCII time-series file
2.
Run TIMESAT on small data sets to define settings
TSM_GUI
3.
Settings file controlling TIMESAT runs
Create settings file
TSM_ settings
4.
Run TIMESAT on full data set
TSF_ process
TIMESAT binary output files
5.
Extract binary images
TSF_ seas2img, TSF_fit2img, etc.
Binary images of seasonality parameters, and ASCII files

Figure 20: *General TIMESAT processing logic. The numbers in the left margin describe the step involved in the data processing. Please refer to the text for further details.*

9 Getting started with TIMESAT – a quick tutorial

This tutorial describes the general steps involved in the processing of time-series data with TIMESAT. The tutorial is meant for getting started with TIMESAT, and more detailed information about the separate functions is given in the reference manual in chapter 10. **The tutorial follows Windows conventions, and Linux users must change the file separator from `\` to `/`.** The sample files for the tutorial are found in the directory `\data`. The subfolder `\data\wa` contains Normalized Difference Vegetation Index (NDVI) data from the NOAA AVHRR sensor over a window covering a part of West Africa. There are also accompanying CLAVR quality data together with a land cover data file `landunits.rst` with three broad classes (cf. section 2.2). This data set will be used in the beginning of the tutorial.

The subfolder `\single` contains single ASCII time-series files for selected areas in Africa and Sweden. The file `MODIS_NDVI_Sweden.txt` covers 9 years with 23 data values per year. There are three time-series in the file. The file `AVHRR_Egypt_82-93.txt` covers 12 years with 36 data values per year. Each year there are two vegetation seasons and there is one time-series in the file. Data in the two ASCII files will be used further on in the tutorial. Basic information about the test data is summarized in the table below.

Directory	**Type**	**Area**	**No. of years**	**Points per year**	**Image dim.**	**Image file type**
`\data\wa`	Image	West Africa	3	36	200×200	8-bit unsigned
`\data\single`	ASCII	Sweden	9	23		
`\data\single`	ASCII	Egypt	12	36		

9.1 Preparing the data

Before running TIMESAT it is necessary to prepare all the input data correctly. TIMESAT expects a data series spanning a number of years, and fits functions to each time-series provided that there is a seasonality pattern in the data.

Input images

TIMESAT needs a sequence of vegetation index images covering a particular geographical area (cf. section 2.2). Images can be downloaded from some data provider e.g. NASA and should be converted to headerless binary format (see sections 9.9, 10.15 and 10.19). The number of images needs to be identical for each year, and each image should represent the same time interval (e.g. one day, 8-days, 10-days, 1 month etc.). If an image representing a certain date is missing, an image denoting missing data should be added. This image should contain numerical values outside the range of the valid data.

Example of vegetation index images are the files named `wa_nd98011.img` etc. provided in the folder `\data\wa`. It is necessary to keep in mind that TIMESAT does not generate output for all the input years. For time-series spanning n years, seasonality parameters will only be extracted for the $n-1$ center most seasons. This limitation is explained in detail under section 4.1. In some cases, when the vegetation season peaks in the middle of the year, the user may duplicate years at the beginning and end of the time-series to get seasonality parameters for all years. The simplest way to do this is just to copy files from a year that is present in the data to new files and add the names of the new files at the beginning or end in the list of input files (see below). Do not forget to update the number of images on the first row!

List of input files

A list giving the total number of files and the full filename and path of each image needs to be present, see sections 5.2, 5.3, and 10.15. An example is the file `ndvilistwa.txt` provided in the `\data\wa` subfolder. Use any ASCII editor to view the file. The file should look like this:

```
108
..\data\wa\wa_nd98011.img
..\data\wa\wa_nd98012.img
      ...
..\data\wa\wa_nd99123.img
```

The first row contains the number of data files (images), then comes one image name (including path) per row. The two dots beginning each path name in the example file mean "up one step" defining a relative path. Thus if you are running from e.g. the `c:\timesat32\run` folder the path will correctly point to the `c:\timesat32\data\wa` folder. However if you are running from another folder (which is not one step down from `timesat32`) the program will not be able to find the files. To be on the safe side you may want to replace the relative path with the full (absolute) path. For Linux users we provide the corresponding file `ndvilistwa_linux.txt`.

In windows, a quick way to generate the image list is to open a command window, navigate to the data folder, and execute the command `dir /b *.img >listfile.txt`. The file, here named `listfile.txt` will need to be opened with a text editor, the no. of images added to the top, the path appended to the file names, and the correct chronological order of the files checked. In Linux the corresponding command is `ls -C1 *.img > listfile.txt`.

Input quality data (optional)

These are images corresponding to each of the vegetation index images. The quality images contain a numerical code that can be used for defining the influence of the image (weight) in the function fitting (cf. section 3.2). A maximum of three numerical intervals can be used for determining the weights, e.g. 1 – 5 to be given weight 1, 6 – 10 to be given weight 0.5 and 11 – 16 to be given weight 0.1. The file format is identical to that of the input vegetation index images, and a separate file list for these quality data should also be present. Example quality images are the files named `wa_cl98011.img` etc. provided in the folder `\data\wa`. We provide a list containing the names of input files in this folder, named `clavrlistwa.txt`. For Linux users we provide the corresponding file `clavrlistwa_linux.txt`.

Single time-series data

An alternative to using images is to extract time-series data for certain pixel locations into an ASCII file, and process these from the file (cf. section 2.1). Several time-series can be processed, and the file format is defined in sections 2.1 and 10.15. It is also possible to store corresponding quality codes in a similar ASCII file. Examples of single ASCII files are provided in the folder `\data\single`.

Land cover data file (optional)

TIMESAT can process data for separate land cover classes (cf. sections 5.2 and 5.3). An image file that assigns a code (0 – 255) to each pixel needs to be present. Each code represents a land cover class. Note that the image format must be the same as for the vegetation index images (see section 9.5). An example is the file `landunits.rst` given in the folder `\data\wa`. It is convenient to keep the land cover file in the same directory as the image data.

9.2 Starting the TIMESAT menu system

The main driver for all TIMESAT processing, Matlab or Fortran, is a menu system. The menu system is divided into three logical areas: data preparation, data processing, and post processing (see Figure 22). To start the menu system please follow the instructions below.

Users with Matlab

First let's make sure that the TIMESAT installation folder is on the Matlab path. Start Matlab and select *Set path* (Figure 21). Click *Add with Subfolders*. Navigate to the folder `timesat_matlab` under the main `timesat33` folder. *Save and Close*. After this, set the TIMESAT working directory to `timesat33\run` (Figure 21). Now, start the menu system `TSM_menu` by typing TIMESAT at the Matlab prompt.

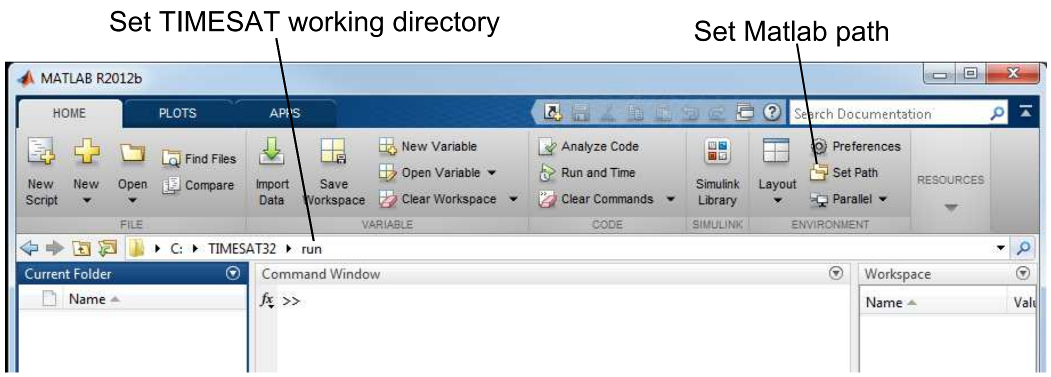


Figure 21: *To start the TIMESAT menu system make sure that the installation folder is on the path, change to the run directory, and type TIMESAT in upper case at the prompt.*

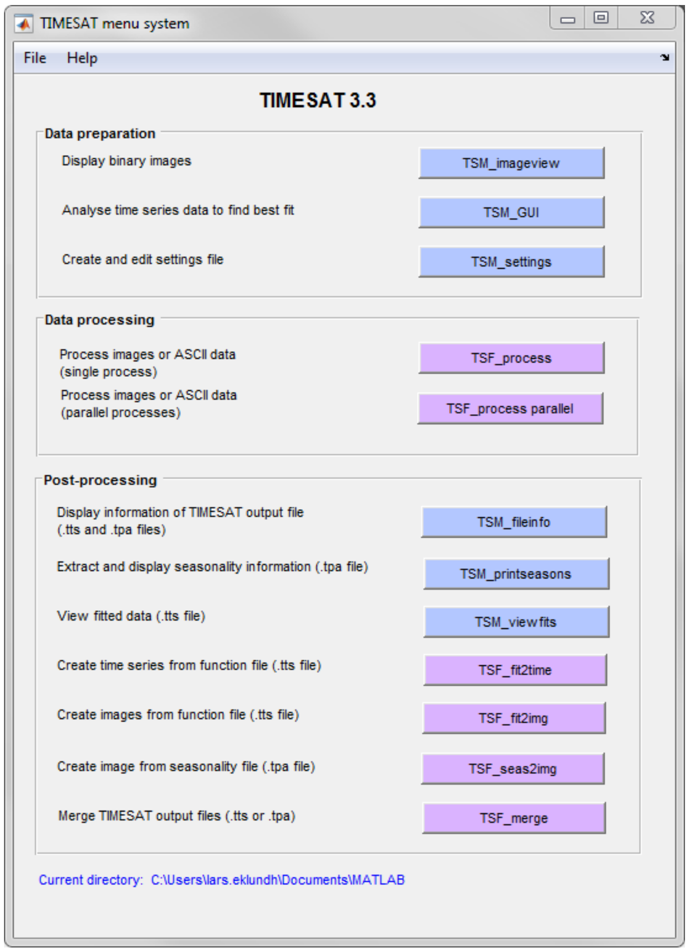

TIMESAT menu system
File Help
TIMESAT 3.3
Data preparation
Display binary images
TSM_imageview
Analyse time series data to find best fit
TSM_GUI
Create and edit settings file
TSM_settings
Data processing
Process images or ASCII data (single process)
TSF_process
Process images or ASCII data (parallel processes)
TSF_process parallel
Post-processing
Display information of TIMESAT output file (.tts and .tpa files)
TSM_fileinfo
Extract and display seasonality information (.tpa file)
TSM_printseasons
View fitted data (.tts file)
TSM_viewfits
Create time series from function file (.tts file)
TSF_fit2time
Create images from function file (.tts file)
TSF_fit2img
Create image from seasonality file (.tpa file)
TSF_seas2img
Merge TIMESAT output files (.tts or .tpa)
TSF_merge
Current directory: C:\Users\lars.eklundh\Documents\MATLAB

Figure 22: *TIMESAT menu system. The system is divided into three logical areas: Data preparation, Data processing, and Post-processing.*

Users without Matlab

Execute the file `TIMESAT.exe` in the `\compiled` folder. This will start up the menu system `TSM_menu` (see Figure 22). Set the working directory by selecting *File* on the menu bar and choosing *Preferences*. Browse to the working directory `timesat33\run` and press *OK*.

9.3 TSM_imageview

Viewing images

We will illustrate this program (see Figure 23) by viewing one of the binary image files provided in the `\data\wa` folder. Start `TSM_imageview` from the TIMESAT menu system. Under *File, Open image file*, browse to the folder `timesat32\data\wa` and click on the `wa_nd99051.img` file. The files contain NDVI data from the NOAA AVHRR sensor (make sure not to select any of the `wa_cl98011.img` files since these only contain quality codes for the images). Leave the choice under *Image file type* to 8-bit unsigned integer. Type 200 under *No of rows in image*, and 200 under *No of columns per row*. These values can be inferred from Table 2. Click the *Draw* button. To modify the *Image display scaling* you can increase the *Minimum* value to about 100 and decrease the *Maximum* value to about 200 (enter these numbers in the edit boxes near the bottom left window corner, or use the sliders). Also try out the other options below the image area: *Zoom on/off*, *Lock axes*, *Grid on/off*, *Datatip on/off*, and *Color scale*.

Browsing through several files

If you have made sure that your file list correctly points to your vegetation index image data (see section 9.1) you may use the function *Open file list* under *File*. Click on the *Open file list* button and browse to `\data\wa` folder and select the `ndvilistwa.txt` file. Click on one of the files, leave the window open and go over to the main window. Choose the correct settings under *Format* and click the *Draw* button. You can then point to another file in the list and just click the *Draw* button again to view this image file.

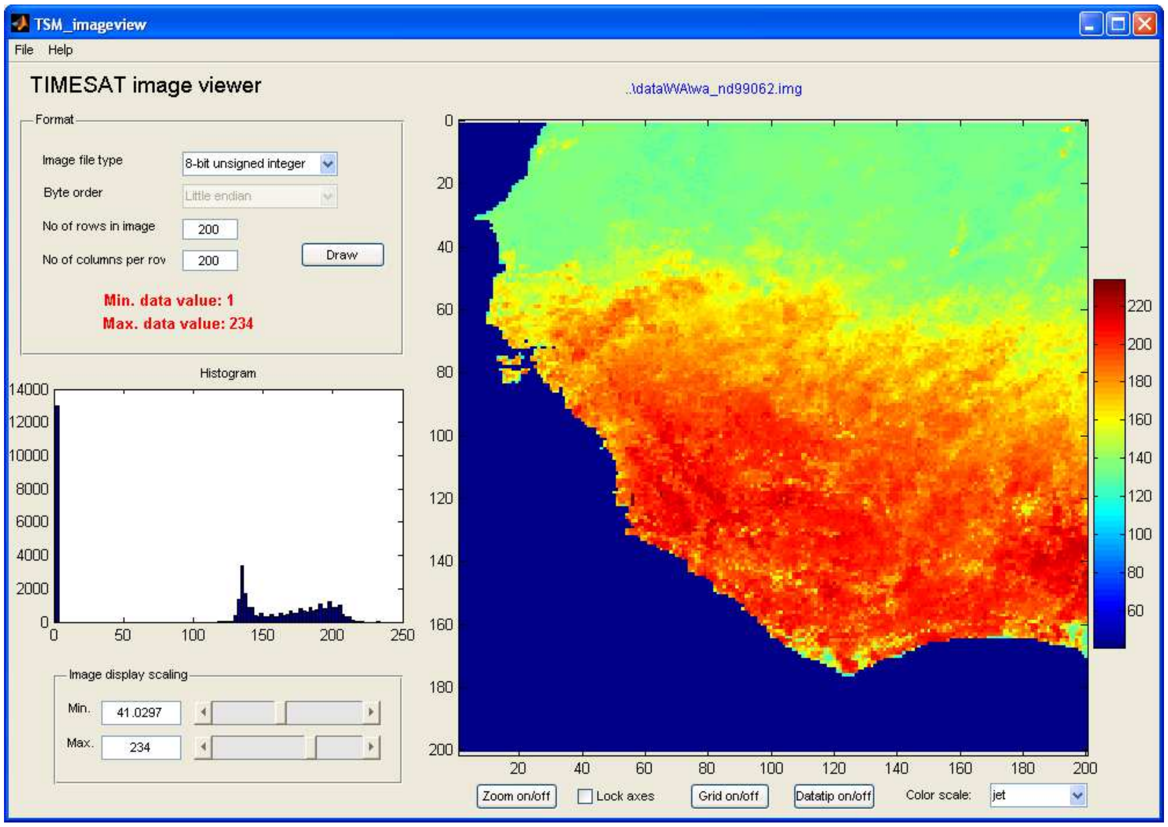


Figure 23: *TSM_imageview with a loaded vegetation index image. The program can be used to view binary image data and explore the correct settings for the file format of images.*

Viewing qualitative land cover data

Use *File*, *Open image file*, browse to the folder `timesat32\data\wa` and click on the file `landunits.rst` and press *Draw*. Under *Color scale*, select *qualitative*. This file contains, as can be explored with the *Datatip* option, a classification of AVHRR NDVI data for 1999 into three broad categories, roughly representing the classes desert (1), semi-arid (2), and semi-humid (3). Note that the classification does not represent an accurate delineation of these eco-climatic regions in West Africa, but was primarily generated for the purpose of this demonstration. When you are done viewing and testing the program you may exit `TSM_imageview`.

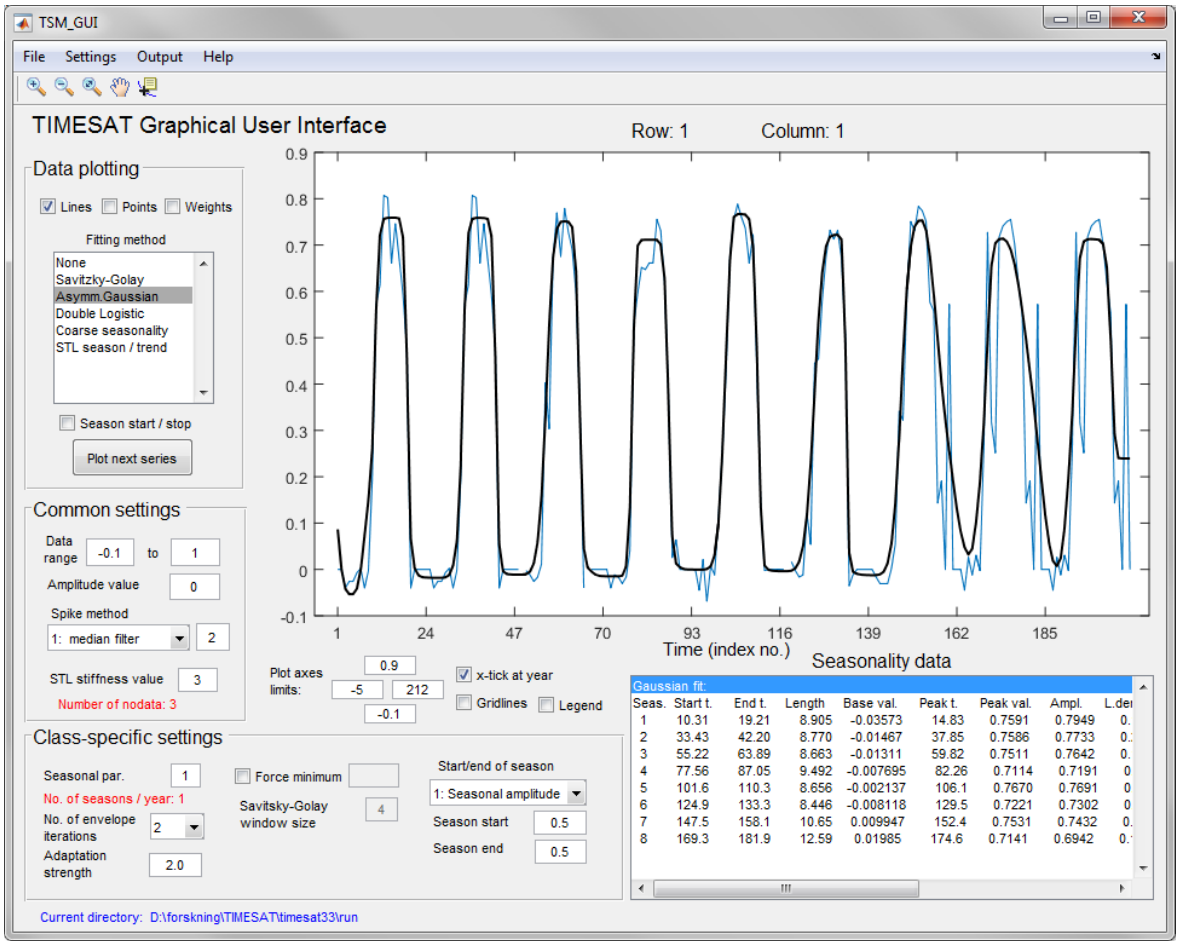
9.4 TSM_GUI

Loading and processing ASCII time-series files

In this example we will load and process data stored in an ASCII time-series file. Click on `TSM_GUI` in the TIMESAT menu system and a window similar to the one displayed in Figure 24 will show up. Then select *File, Open ASCII data file*. Use the *Browse* button to open the file `\data\single\MODIS_NDVI_Sweden.txt`. This file contains NDVI data from MODIS for the time period 2000 – 2008. Note the preview of the file contents loaded into the window. The first row shows that there are 9 years of data, 23 observations per year, and 3 time-series. Press *Load data*. The raw data from the first row of the file will load into the plotting area of `TSM_GUI`.

Next, choose the different options under *Data plotting*. Note the different fits achieved with *Gaussian*, *Logistic* and *Savitsky-Golay*. More than one fitting method can be displayed simultaneously by holding the `ctrl-key` while selecting method with the mouse. The fits are affected by a number of options for detecting spikes, adapting to the upper envelope etc. These options can be controlled by choices (check boxes and buttons) in the GUI either under *Common settings* or *Class-specific settings*. The different options are discussed in detail in section 5.3. When Savitsky-Golay is selected you may change the *Savitsky-Golay window size* under *Class specific settings* (cf. sections 3.6 and 5.3). Try a large and small window size. Note that the change of the window size takes effect when you hit enter or click outside the editing area. There are more options, including the *Spike method*, *Number of envelope iterations* and *Adaptation strength*, that you might want to explore.

When any of the fitting methods is selected, seasonality parameters for each of the defined seasons are shown under *Seasonality data* to the right. More information about these parameters is given in section 4.4. To write the parameters to file select the *Output* menu and choose *Write seasonality data to file*. Now a file named `seasonality.txt` will be found in your working directory. You can open this file with any text editor. Note that the number of seasons equals $n-1$ in case of single seasons and $2n-1$ in case of dual seasons, where n is the number of years. You can change the definition of the beginning and end of seasons. Make sure that at least one fitting method is selected, and select *Season start/stop* under *Data plotting*. Keep the *Start of season method* (under *Class specific settings*) at 1 and modify the *Season start* and *Season stop settings*. These settings can be varied between 0 and 1 and define the fraction of the amplitude used for determining the beginning and end of the seasons (cf. section 5.3). After that you can test choosing 2 under *Start of season method*.



Data range -0.1 to 1
Amplitude value 0
Spike method 1: median filter 2
STL stiffness value 3
Number of nodata: 3

Seasonal par. 1
No. of seasons / year: 1
No. of envelope iterations 2
Adaptation strength 2.0
Force minimum
Savitsky-Golay window size 4
Start/end of season 1: Seasonal amplitude
Season start 0.5
Season end 0.5

Plot axes limits: 0.9, -5, 212, -0.1

Current directory: D:\forskning\TIMESAT\timesat33\run

Gaussian fit:

Seas.	Start t.	End t.	Length	Base val.	Peak t.	Peak val.	Ampl.	L.de
1	10.31	19.21	8.905	-0.03573	14.83	0.7591	0.7949	0.
2	33.43	42.20	8.770	-0.01467	37.85	0.7586	0.7733	0.
3	55.22	63.89	8.663	-0.01311	59.82	0.7511	0.7642	0.
4	77.56	87.05	9.492	-0.007695	82.26	0.7114	0.7191	0
5	101.6	110.3	8.656	-0.002137	106.1	0.7670	0.7691	0
6	124.9	133.3	8.446	-0.008118	129.5	0.7221	0.7302	0
7	147.5	158.1	10.65	0.009947	152.4	0.7531	0.7432	0.
8	169.3	181.9	12.59	0.01985	174.6	0.7141	0.6942	0.

Figure 24: *TSM_ GUI showing NDVI over West Africa, row 91, column 111. The asymmetric Gaussian model fit is displayed along with the raw data values.*

This option means that an absolute y-value that defines the beginning and end of seasons will be used. Select 2 under *Start of season method*, and modify the values under *Season start and Season stop*. Try values between 0 and 0.65 to see the effect.

Please note that the check boxes and buttons in the GUI correspond to the input settings described in 5.3. Some of the input settings, such as the number of years and the number of observations per year, are defined already when data are loaded. As we will describe later the selected options can be transferred to the settings file.

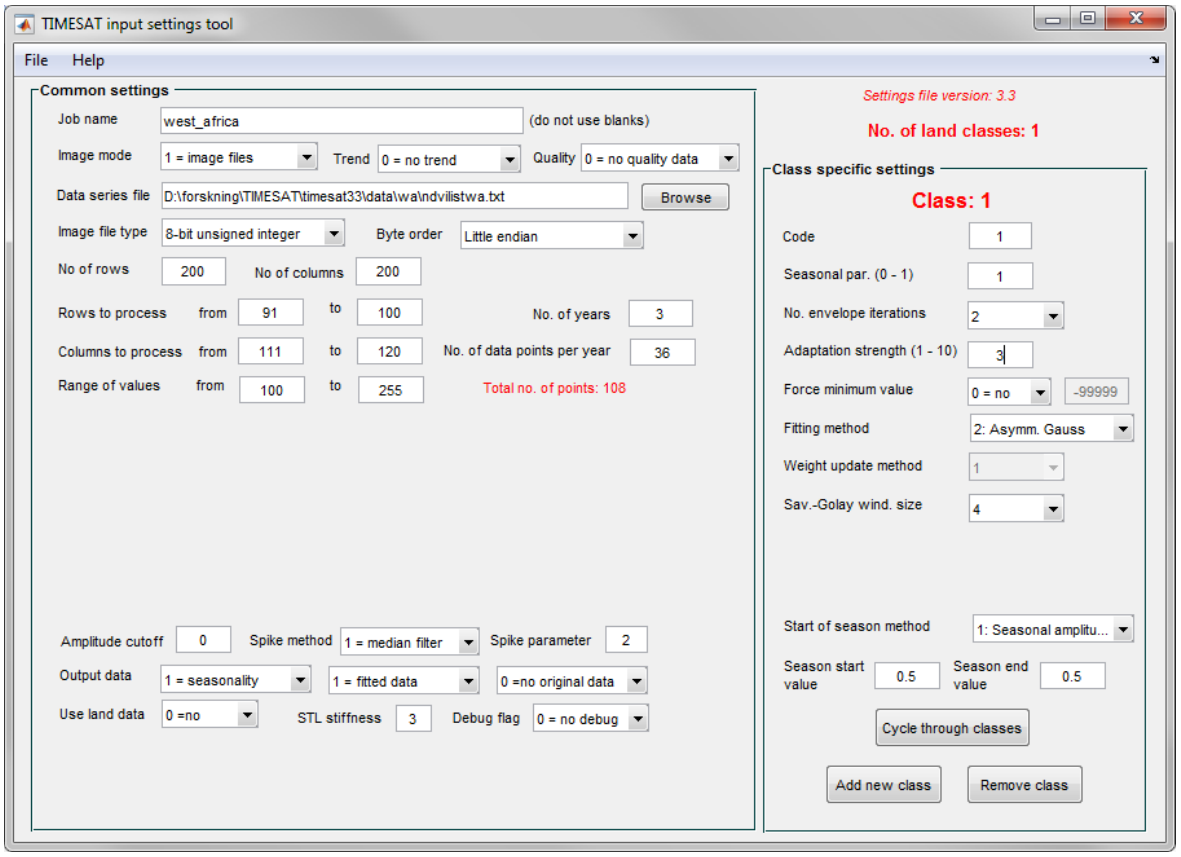
Loading and processing binary vegetation index image data

In this example we will load and process data stored in binary vegetation index images. These are NDVI 10-day composites from NOAA AVHRR over a section of West Africa. Make sure `TSM_GUI` is started, and choose *File*, *Open list of image files*. This will ask you to select a file list with names of binary images in sequence, starting with the first and ending with the last (cf. section 9.1). Use the *browse button* and choose the file `ndvilistwa.txt` under the `\data\wa` folder. Fill in 3 under *No. of years*. Press <tab> to move to the next item to be filled. You will note that the value 36 ($nptperyear = nts/nyear$) was automatically filled in. Tabulate to *No. of rows in image* and fill in 200, then to *No. of columns per row* and fill in 200. Press *Show image*. This is not mandatory, but allows you to preview the images in the file list and to select a processing window. Press the button *Processing window*, which gives a movable hair cross, and select a small area of approximately 10 × 10 pixels roughly at locations rows 91–100 and col 111–120. Press *Return* when you are done. Back in the `image_files_input` window you will now see that the selected window coordinates have been filled into edit boxes *Rows to process* and *Columns to process*. If you are not satisfied you can modify these coordinates to the ones above manually. Now press *Load data*. Select *Gaussian fit* under *Data plotting*. You should see a curve like the one in Figure 24.

Common settings. These settings affect all pixels in the images, irrespective of class (cf. sections 5.2 and 5.3). The first to set is *Data range*. Set the range from 1 to 255 since these are the minimum and maximum values of the the data used in this example. All values outside this interval will be ignored in the processing of full imagery. You may try to set this to a more narrow interval, e.g. 170 – 255, but too few pixels will be available for processing and the pixel will be skipped. Set the values back to 1 to 255. Then try increasing *Amplitude value* from 0 to a higher value. At about 39 and higher you should obtain an error message. This threshold value can be used to remove pixels with very weak seasonality from the processing, e.g. desert areas. Change the value back to zero to make sure you include all pixels in the processing. Next setting to try is *Spike method*. Select method 1 to remove single values that deviate more then a specific distance from the running median. A value of 2 denotes that spikes larger than two standard deviations from the running median will be removed. The value can be decreased to remove smaller spikes, or increased, to allow larger spikes. Also try methods such as STL replace.

Class-specific settings. These settings are specific to individual land classes (cf. sections 5.2 and 5.3). `TSM_GUI` only recognizes one land class, but when running whole images using `TSF_process`, it is possible to process data individually for different land classes. The first class-specific setting is the *Seasonal parameter*. Set this to 1 since we assume a single season per year (and to 0 if you assume dual seasons per year). We will explore this more in another example. Then go on to *No. of envelope iterations*. When it is set to 1 no envelope adaptation is carried out, and when set to 3 maximum adaptation is done (cf. section 3.4). This can be further fine tuned using *Adaptation strength*. Minimum strength is 1 and maximum is 10. Try different values, then set it to 2 and *No. or envelope iterations* to 3. The last setting to discuss is *Force minimum*. This setting can be used to force minimum values to a certain value. Check the box and enter a value of 160 to see the effect. All values below 160 will now be forced to 160. Uncheck the box for the remainder of this demonstration. The rest of the settings under *Class specific values* have been discussed above. Now, go through a number of pixels by clicking the button *Plot next series* under *Data plotting* and try to find common and class specific settings that make the curves fit nicely to the data. This is more of an art than a science and has to be made with due consideration to the nature of the ground target and type of satellite data used.

Saving settings to file. To save the settings currently selected choose *Settings, Save to settings file*. This will start the tool `TSM_settings` and parse all the current `TSM_GUI` settings



Common settings

Job name: west_africa (do not use blanks)
Image mode: 1 = image files; Trend: 0 = no trend; Quality: 0 = no quality data
Data series file: D:\forskning\TIMESAT\timesat33\data\wa\ndvilistwa.txt
Image file type: 8-bit unsigned integer; Byte order: Little endian
No of rows: 200; No of columns: 200
Rows to process: from 91 to 100; No. of years: 3
Columns to process: from 111 to 120; No. of data points per year: 36
Range of values: from 100 to 255; Total no. of points: 108
Amplitude cutoff: 0; Spike method: 1 = median filter; Spike parameter: 2
Output data: 1 = seasonality; 1 = fitted data; 0 =no original data
Use land data: 0 =no; STL stiffness: 3; Debug flag: 0 = no debug

Settings file version: 3.3
No. of land classes: 1

Class specific settings

Class: 1

Code: 1
Seasonal par. (0 - 1): 1
No. envelope iterations: 2
Adaptation strength (1 - 10): 3
Force minimum value: 0 = no; -99999
Fitting method: 2: Asymm. Gauss
Weight update method: 1
Sav.-Golay wind. size: 4
Start of season method: 1: Seasonal amplitu...
Season start value: 0.5; Season end value: 0.5

Figure 25: *TSM_settings with settings for West Africa. Common settings are shown in the left pane and class specific settings in the right pane. For more information please see text.*

to the tool (Figure 25). Common settings are found in the left frame and class specific settings in the right frame. In addition there are some more settings related to e.g. the type of output data (see sections 6.1 and 6.3) and land classes that may be chosen. Go to *Job name* and enter `west_africa`. This is an important setting that determines the names of all output files when running `TSF_process`. Then go to *Output data* and set *seasonality*, *fitted data*, and *original data* all to 1 so that the files `.tpa` and `.tts` will be printed. Now save the settings to an output file by going to *File, Save settings file as*. Type the name `west_africa` and click *Save*. Then exit `TSM_settings` and return to the `TSM_GUI` window. You may now view the new file `west_africa.set`, which resides in the run directory, with any file editor.

Loading settings from file. To demonstrate how settings are loaded from file you can now exit `TSM_GUI` to clear all settings. Then open it again from `TSM_menu`. Load the settings file saved in the previous step by going to *Settings* and choosing *Load settings file*. Browse to the file `west_africa` and click on *Open*. The data and settings specified in the file `west_africa.set` will now be loaded.

Managing data with dual seasons

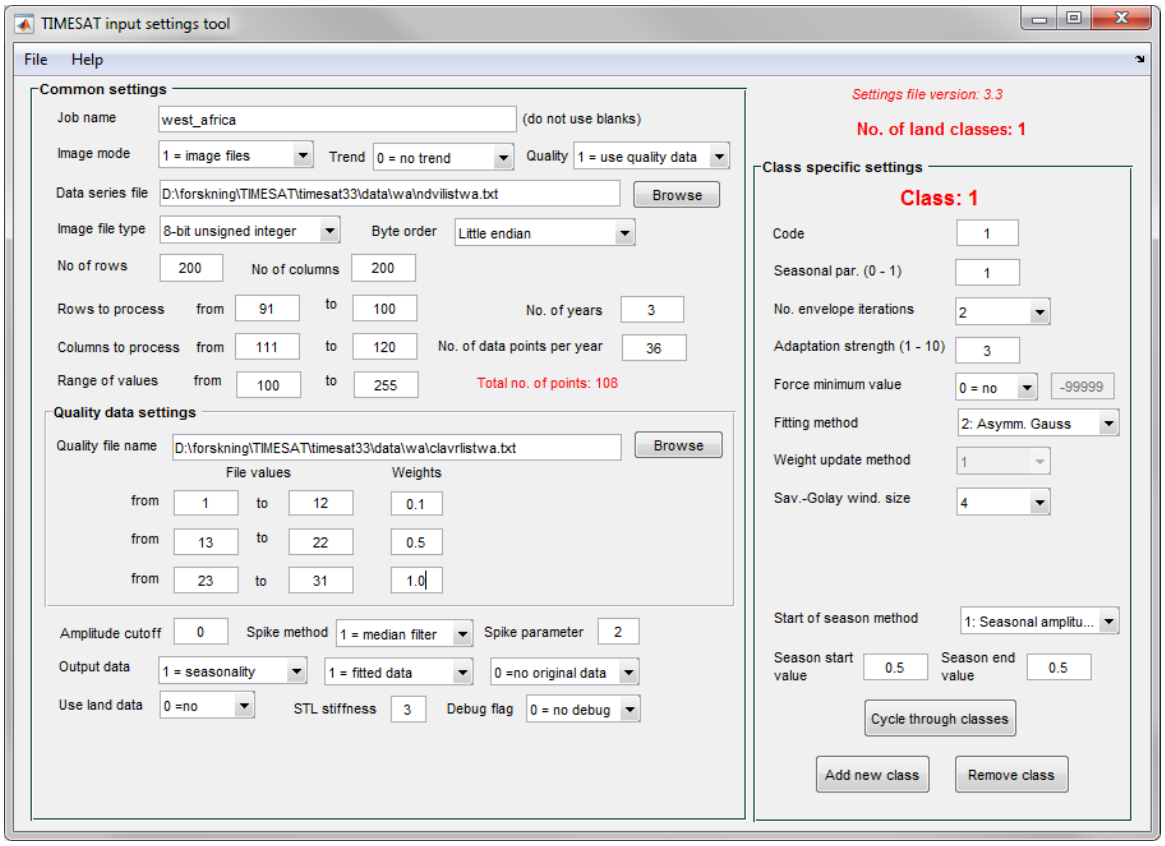
In `TSM_GUI` go to *File, Open ASCII data file*. Browse to the `\data\single` directory under TIMESAT, and select the file `AVHRR_Egypt_82-93.txt`. This ASCII file contains 12 years of 10-day NDVI-composite data (scaled values) from the NOAA AVHRR sensor over a point in the Nile valley. *Load* the data and select *Logistic fit*. With the default setting of the *Seasonal parameter* (under *Class-specific settings*) of 0.5, the logistic curve will not fit well to the data. If you check *Coarse seasonality* the resulting curve of the initial coarse fit is shown (see section 3.5). Now, change the *Seasonal parameter* to 0 to force the seasonality to dual seasons instead of a single season. Both the coarse fit and the logistic fit will now match the data better. To improve the fit you may need to zoom in on a few years. You may then try different settings of *No. of envelope iterations* and *Adaptation strength*.

Weighting observations with quality data

Load data for West Africa using the previously saved settings file `west_africa.set` (*Settings, Load settings file*. Go to *File, Open list of image files* and check *Use quality data*. We will now use quality indices from the CLAVR (Clouds from AVHRR) data set to weight the function fitting in TIMESAT (see section 3.2). The quality indices are stored in binary image files of the same data type and format as the NDVI image data. Use the *Browse* button next to the *Weight list file*, and open the file `clavrlistwa.txt` in the `\data\wa` directory. Under *File values* type the following

File values				Weight
From	1	to	12	0.1
From	13	to	22	0.5
From	23	to	31	1.0

The above values will assign the weights in the right column to the data ranges under file values. We will here assign the lowest weight to values between 1 and 12 (cloudy). We use the weight 0.1 rather than 0.0 to the lowest class to avoid possible problems that might arise if long sequences of data of low quality occur. For the middle class (mixed) we assign weight 0.5 to values between 13 and 22. For the highest class (clear) we assign weight 1.0 to values between 23 and 31. Then press *Load data*. The chosen weights 0.1, 0.5 and 1.0 are arbitrarily chosen and represent the relative influence we want to assign each of the categories in the data fitting. Back in the `TSM_GUI` window check *Weights* under *Data plotting* to see which data points are assigned what weights. A pixel where the effect of using or not using weighting with the above values will have a strong effect is row 111, column 117. Load this pixel (*File, Open list of image files*) and try varying the weight values to see what the effect will be on the fitted curve. Reset the data to the above values and save the new settings to `west_africa.set` by going to *Settings, Save to settings file*. In `TSM_settings` save the file and overwrite the old `west_africa.set`.


TIMESAT input settings tool

File Help

Common settings

Job name: west_africa (do not use blanks)
Image mode: 1 = image files
Trend: 0 = no trend
Quality: 1 = use quality data
Data series file: D:\forskning\TIMESAT\timesat33\data\wa\ndvilistwa.txt
Image file type: 8-bit unsigned integer
Byte order: Little endian
No of rows: 200
No of columns: 200
Rows to process from 91 to 100
No. of years: 3
Columns to process from 111 to 120
No. of data points per year: 36
Range of values from 100 to 255
Total no. of points: 108

Quality data settings

Quality file name: D:\forskning\TIMESAT\timesat33\data\wa\clavrlistwa.txt

File values				Weights
from	1	to	12	0.1
from	13	to	22	0.5
from	23	to	31	1.0

Amplitude cutoff: 0
Spike method: 1 = median filter
Spike parameter: 2
Output data: 1 = seasonality; 1 = fitted data; 0 =no original data
Use land data: 0 =no
STL stiffness: 3
Debug flag: 0 = no debug

Settings file version: 3.3

No. of land classes: 1

Class specific settings

Class: 1

Code: 1
Seasonal par. (0 - 1): 1
No. envelope iterations: 2
Adaptation strength (1 - 10): 3
Force minimum value: 0 = no; -99999
Fitting method: 2: Asymm. Gauss
Weight update method: 1
Sav.-Golay wind. size: 4
Start of season method: 1: Seasonal amplitu...
Season start value: 0.5
Season end value: 0.5

Figure 26: *TSM_settings with quality data.*

9.5 TSM_settings

Modifying the settings file

We will now look more closely at the program for modifying settings files. Start `TSM_settings` from the TIMESAT menu system. Recall that we had saved a file called `west_africa.set` in the current run directory. Load this file by going to *File*, *Open settings file*, and select `west_africa.set`. The window should look like the one in Figure 26. We will now modify some of the settings to prepare for processing the full image data set using multiple land classes. Go to *Rows to process* and *Columns to process* and change each of these so that the processing is from 1 to 200 instead. Set *Use land data* to 1 and click the *Browse* button to the right of *Land cover file* and choose the file `landunits.rst` in the `\data\wa` folder.

Working with multiple land classes

We will now add land classes for each of the land units shown in the map `landunits.rst` displayed under 9.3. In this map the following codes were used: 0: water, 1: desert, 2: semi-arid, and 3: semi-humid. Let's say that the settings we decided on earlier fit well in the class 2 above, we will need to add settings for the other land units as well. First, let's assign a land code to the class that we have already defined. Go to *Code* under *Class specific settings*, and change it to 2. Then add a new class by clicking on the button *Add new class*. Let's add settings for the desert areas. Change *Code* to 1, *Seasonal par.* to 1, *No. of envelope iterations* to 2, *Adaptation strength* to 2, *Fitting method* to 2, *Value for season start* and *Value for season stop* each to 0.3. Then add another class, representing the semi-humid areas. Set *Code* to 3, *Seasonal par. to 1*, *No. of envelope iterations to 3*, *Adaptation strength to 5*, *Fitting method to 1*, *Sav-Golay wind. size to 5*, and *Value for season start* and *Value for season stop* to 0.5. Do *not* press *Add new class* again. The Ocean areas (code = 0) should not be processed, so we will leave this classed undefined. Use the button *Cycle through classes* to check the settings for each of the classes. You have the option to *Remove class* if you have accidentally added a class too many. Save the file under the name `west_africa_full.set`. When you run `TSF_process` with this file, different settings will be applied to each of the land cover units in the file `landunits.rst` (see section 2.2).

9.6 TSF_process

This program carries out data smoothing and function fitting for all pixels in an image stack. It runs as single or multiple (see 9.7) processes, controlled by a settings file.

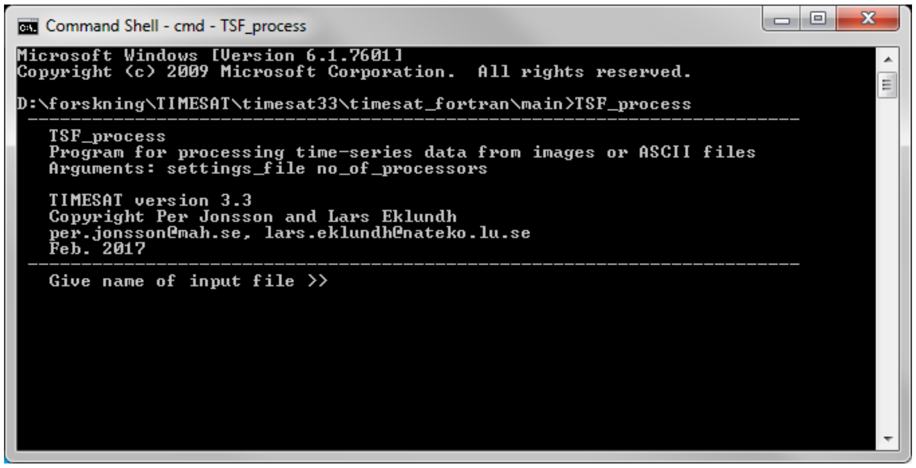
To start the program click on `TSF_process` in the TIMESAT menu system. TIMESAT will ask for the input settings file. Select the file `west_africa_full.set`. A command window will then open and `TSF_process` will start running immediately. You can also start `TSF_process` directly from a separate command window by typing `TSF_process`. If this window has not been opened by TIMESAT it is necessary to first set the path to the Fortran executables. In DOS this is done by typing a similar command to the following at the command prompt:

```
set path=c:\timesat33\timesat_fortran\main;c:\timesat33\timesat_fortran\tools;%path%
```

In Linux it is done by typing the following command:

```
PATH=/myhome/timesat33/timesat_fortran/main:/myhome/timesat33/timesat_fortran/tools:$PATH
```

The actual path on the user's own computer to the TIMESAT executable directories should replace those in the examples above. After starting `TSF_process` directly from a separate command window the message *Give name of input file:* will appear on the command line (note that here you will need to use the full file name, including the file prefix `.set`). Type the name of the settings file and press return. The question Give number of processors will appear. Answer 1 to this, and the processing will start (see next session for running multi-processor jobs). Alternatively, you can use the command line format and launch the program by writing `TSF_process` followed by the name of the settings file and 1 on the same line and pressing return. The command window is shown in Figure 27.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

D:\forskning\TIMESAT\timesat33\timesat_fortran\main>TSF_process
 ------------------------------------------------------------------------
   TSF_process
   Program for processing time-series data from images or ASCII files
   Arguments: settings_file no_of_processors

   TIMESAT version 3.3
   Copyright Per Jonsson and Lars Eklundh
   per.jonsson@mah.se, lars.eklundh@nateko.lu.se
   Feb. 2017
 ------------------------------------------------------------------------
   Give name of input file >>
```

Figure 27: *Command window for running TSF_process started from the command prompt.*

The program will process the data row by row. As you will notice, when having large data files the processing can take considerable time.

If you are not prepared to wait for the finished result we recommend you to stop the processing (hit *crtl-c*), and decrease the data window to be processed. This can be done by editing the settings file directly or by using the `TSM_settings` program. To be able to continue this tutorial in a meaningful way we recommend that you make the window relatively large, e.g. Rows: 1 – 100, columns: 1 – 60. To run the program again, type `TSF_process` in the DOS command window. It should not take many seconds to complete the run with the modified file on an ordinary PC. After the run the command window should look like this:

```
      ....
 Row           99
 Row          100

 Index file west_africa_TS.ndx created

 Index file west_africa_fit.ndx created

  -- Processing finished --

 Seasonality parameters written to:
          west_africa_TS.tpa
 Fitted functions written to:
          west_africa_fit.tts
 Date started: 20121112
 Time started: 115559.066
 Date stopped: 20121112
 Time stopped: 115602.086
```

Data from the processing are saved to file. Check the contents of your current working directory and note that the two files `west_africa_TS.tpa`, `west_africa_fit.tts`, and corresponding index files (`.ndx`) have been created (cf. sections 6.1 and 6.2).

9.7 TSF_process parallel

This executes `TSF_process` on multiple parallel runs, thereby reducing the processing time. The parallel processing is done in two steps. In the first step data are analyzed and a script file is created that assigns the job to the different processors. The assignment is done is such a way that a good load balance is obtained. In the second step the script file is run and the work is distributed on the different processors. After each processor has finished the output files are merged to single files and intermediate files are removed. To initiate the first step click on `TSF_process parallel` in the TIMESAT menu system. TIMESAT will ask for the input settings file. Select the file `west_africa_full.set`. The message *Enter no. of processors* will appear. Enter the number of processors of you system, e.g. 4, and press OK and the analysis of the data will start. At the end of the analysis the script file `west_africa_script.bat` (Windows) or `west_africa_script.sh` (Linux) is produced. To start the multi-processor job type `west_africa_script.bat` (Windows) or `./west_africa_script.sh` Linux on the command line. Please note that for the script file to work the Fortran executables *must* be on the path (see section above). Parallel processing is not entirely straight forward and may depend on the operating system. In case of problems the user is advised to read the detailed description of the parallel processing found in section 10.9. After the run the command window should look like this:

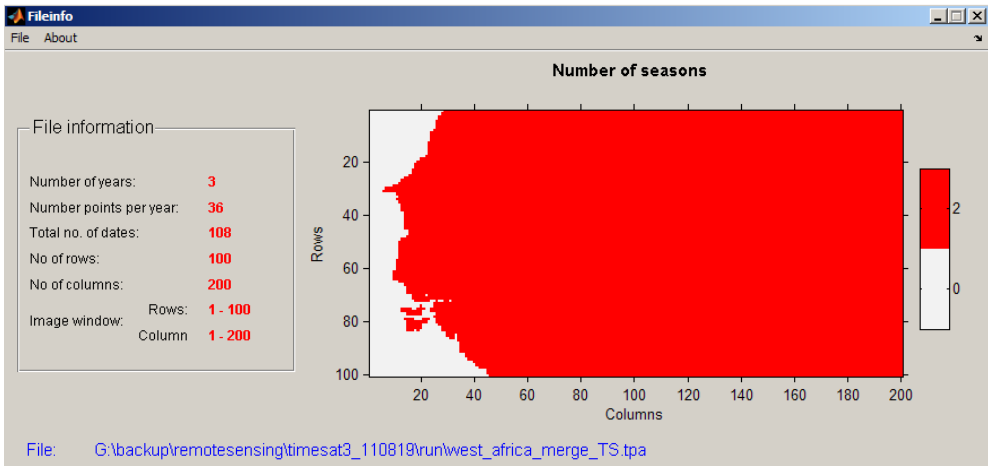
```
       ....
Parallel jobs finished
Merging files
 Done merging files
Cleaning up
 Done merging files
Cleaning up
Processing done
```

9.8 Post-processing the results of a TSF_process run

The main TIMESAT processing is now done and we will see how data can be post-processed and viewed (cf. section 6.3). We provide a few basic programs for opening and processing the output files, but for certain processing the user may be required to create tailored software.

TSM_fileinfo

This routine provides information about a TIMESAT output file. To activate it you click on `TSM_fileinfo` in `TSM_menu`. You will be asked to select an input file. Select `west_africa_TS.tpa`. A window will open that provides the following information: no. of years, no. of points per year, total no. of points, no. of rows and columns, extent of processing window. When the output is from image mode processing a map showing the number of seasons for each pixel is displayed to help the user to geographically localize the area that has been processed (Figure 28).



File information

Number of years: 3
Number points per year: 36
Total no. of dates: 108
No of rows: 100
No of columns: 200
Image window: Rows: 1 - 100
Column 1 - 200

File: G:\backup\remotesensing\timesat3_110819\run\west_africa_merge_TS.tpa

Figure 28: *File information along with a map to localize the processed area.*

If you select a file containing fitted data (e.g. `west_africa_fit.tts`), the map will display the number of years with fitted data available in the output file.

TSM_printseasons

Program for viewing the seasonality parameters for selected pixels or points. This is the same information as generated in `TSM_GUI`, but more convenient for printing information for a large number of pixels to file. Press `TSM_printseasons` in `TSM_menu` (you can also start the program directtly from the Matlab prompt). The program will start in the Matlab command window, and ask you to specify a `.tpa` file and a specific area to view. Give the name `west_africa_TS.tpa`. To view seasonality information for pixels between rows 30 and 35 and columns 50 and 50 complete the following dialogue:

```
--------------------------------------------------------------
   TSM_printseasons
   Reads the output file from the TIMESAT program
   and prints seasonality data

   TIMESAT version 3.3
   Copyright Per Jonsson and Lars Eklundh
   per.jonsson@mah.se, lars.eklundh@nateko.lu.se
   Feb. 2017
--------------------------------------------------------------
   Give name of seasonality file:
>> west_africa_TS.tpa
   Data window in file
  -------------------
   Rows    : 1 - 200
   Columns : 1 - 200

   Now enter the window you wish to display data for
   First row:
>> 30
   Last row:
>> 35
   First column:
>> 50
   Last column:
>> 50

   Name of output text file (hit Enter to print to screen):

   Row, Column:  30 50
   Data for season 1
    Beg.    End.   Length   Base   Mid-x   Max.    Amp.    L-der.
    22.2    32.4    10.2   142.6    26.8   188.0    45.4     7.5  ...
   Data for season 2
    Beg.    End.   Length   Base   Mid-x   Max.    Amp.    L-der.
    57.5    68.0    10.5   141.9    62.2   194.5    52.6     9.9  ...
```

```
 Hit enter to continue
```

This shows you seasonality parameters for the first pixel in the specified window. Please note that the parameters are settings dependent and that you may not get the same values as displayed here in the manual. TIMESAT has identified two seasons, i.e. number of years −1 (see section 4.1). The beginning of the first season is at 22.5. In this case, the time step is 10 days, meaning that it occurs at day 225. The end is at 32.0, i.e. day 320. The length of the first season is 9.5, i.e. 95 days. Season two starts at 57.6 (day 576 from the beginning), and ends at 67.2 (day 672), and is 9.6 (96 days) long. See section 4.4 for explanation of the remaining seasonality parameters. If you continue to hit *Enter* you will see seasonality information for the remaining pixels in the window. When specifying an output file all the information will be printed to an ASCII file in your working directory.

TSM_viewfits: viewing the fitted functions

This Matlab program lets you view fitted data stored in the TIMESAT `.tts` files. To view original time-series and fitted functions for pixels between rows 30 and 35 and columns 50 and 50 complete the dialogue below:

```
-----------------------------------------------------------------------
 TSM_viewfits
 Reads the output function files from the TIMESAT program
 and displays fitted functions and original time-series

 TIMESAT version 3.3
 Copyright Per Jonsson and Lars Eklundh
 per.jonsson@mah.se, lars.eklundh@nateko.lu.se
 Feb 2017
-----------------------------------------------------------------------

 Display fitted functions?  (1/0)?
>> 1
 Display original time-series?  (1/0)?
>> 1
 Give name of input fitted data file:
>> west_africa_fit.tts
 Give name of input raw data file:
>> west_africa_raw.tts

 Data window in file
 -------------------
 Rows    : 1 - 100
 Columns : 1 - 60

 Now enter the window you wish to display data for
 First row:
```

```
>> 30
 Last row:
>> 35
 First column:
>> 50
 Last column:
>> 50
 Displaying data for row: 30 column 50
 Hit enter to continue
```

A plot window will open with fitted data for the first pixel. By hitting *Enter* you can view the remaining pixels in the window. This program is valuable for systematically checking fits of a larger run.

TSF_fit2time: Writing fitted time-series data to an ASCII file

Select `TSF_fit2time` and you will be asked to select an input `.tts` file (`raw` or `fit`). Here we choose `west_africa_fit.tts`. A command window will open with the dialogue:

```
 -----------------------------------------------------------------------
 Program TSF_fit2time
 Program for generating ASCII time-series from TIMESAT fitted
 or raw data files
 Arguments: infile frow lrow fcol lcol outfile

 TIMESAT version 3.3
 Copyright Per Jonsson and Lars Eklundh
 per.jonsson@mah.se, lars.eklundh@nateko.lu.se
 Feb 2017
 -----------------------------------------------------------------------

 No. of years:     3,  No. of points per year:    36
 Image window (rows, cols):     1 -   100 ;     1 -    60

 Specify image window to extract data for
 (first row , last row, first column, last column):
>> 30 35 50 50
 Name of output file:
>> timeseriesfile.txt
```

Fitted data (original time-series data, if we are reading a `raw` file) have now been written to the file `timeseriesfile.txt`. The data structure is the same as the one described in section 2.1. The first line gives the number of years, the number of data values per year, and the total number of time-series in the file. After that the extracted time-series are given row by row: all time-series belonging to row 30 then all those belonging to row 31 etc. You can view and edit the output file with your file editor. The file should look like this:

```
% Columns: row, col, data. No. of years: 3. No. of points per year: 36. No. of series:6
```

```
30     50  1.49521E+02  1.46688E+02  1.44439E+02  1.42741E+02   ...
31     50  1.62171E+02  1.56509E+02  1.51780E+02  1.47949E+02   ...
...
```

where the first two values on each row are the row and column numbers. Data are displayed in scientific notation.

TSF_fit2img: Creating an image from the fitted data

This program generates an image from the smoothed data generated by the fitting methods *Gaussian*, *Logistic* or *Savitzky-Golay*. Activate `TSF_fit2img` and again give the name of the fitted data file. Follow the dialogue:

```
 -------------------------------------------------------------
 Program TSF_fit2img
 Program for generating images from TIMESAT fitted data files
 Arguments: infile misspix outfile filetype image_no

 TIMESAT version 3.3
 Copyright Per Jonsson and Lars Eklundh
 per.jonsson@mah.se, lars.eklundh@nateko.lu.se
 Feb 2017
 -------------------------------------------------------------

 Code to give to missing pixels
>> 0
 Name of output files (no extension!)
>> west_africa_fit

 Specify the file type for the output image files (1,2 or 3)
 1 = 8-bit binary
 2 = 16-bit signed integer
 3 = 32-bit real
>> 3

 No. of years:    3,  No. of points per year:   36
 Image window (rows, cols):    1 -  100 ;    1 -   60

 Image no. to map (-1 = all)
>> 54
 west_africa_fit_0054 opened
 Running...
 Fitted data images written
 File format 32-bit real
```

In this example we choose to create an image from the 54*th* time step. Since there are 36 images per year it follows that this is in the middle of the second year ($54 - 36 = 18$). We have chosen to give missing data pixels (e.g. those where TIMESAT has not fitted any data) value 0. The output data has been formatted to 32-bit real for maximum precision. To create images for all the dates specify −1 under *Image no. to map*.

To view the created image, activate `TSM_imageview` and load the file `west_africa_fit_0054`. Note that the file format is 32-bit real, and that the size of the file is 100 rows × 60 columns.

TSF_seas2img: Creating an image from the seasonality data

This program generates an image from the seasonality parameters generated by TIMESAT. Click on `TSF_seas2img` in `TSM_menu`. Now enter the seasonality file `west_africa_TS.tpa`. Then specify the seasonality parameter to map (see section 4.4). Here we will map the *Start of season* (1). Then specify an interval that is wide enough to cover the second season (see section 6.3). A suggestion is to specify this interval to 30 to 82 since this will overlap the second year (which is contained in images 37 – 72). Define codes for missing data due to no season found within the interval, and no pixel data at all found at that location. Here we will separate them by giving different codes, 0 and 1. Finally, give a name of the output file, and specify its format. Carefully note the format since it is important when viewing the file with `TSM_imageview`. Here we will give the name `begin`, and specify output in full precision, 32-bit real. The full dialogue is shown below.

```
 ------------------------------------------------------------------------
   Program TSF_seas2img
   Program for generating images from TIMESAT seasonality files
   Arguments: infile seaspar datemin datemax misseason misspix nameout filetype

   TIMESAT version 3.3
   Copyright Per Jonsson and Lars Eklundh
   per.jonsson@mah.se, lars.eklundh@nateko.lu.se
   Feb. 2017
 ------------------------------------------------------------------------

   Start-of-season time _________________1
   End-of-season time ___________________2
   Length of season _____________________3
   Base value ___________________________4
   Time of middle of season _____________5
   Maximum value value of fitted data ____6
   Amplitude ____________________________7
   Left derivative ______________________8
   Right derivative _____________________9
   Large integral ______________________10
   Small integral ______________________11
   Start-of-season value _______________12
   End-of-season value _________________13

   Seasonal parameter to output
```

```
>> 1
   Specify time interval between which the season MAXIMUM is expected to occur.
   Only pixels with seasons that have a maximum between given dates
   will be processed.
   Example: to extract the seasons of the second year using monthly data
   (i.e. 13-24), specify a min value of e.g. 13 and a max value of e.g. 24.
   Give min,max dates(e.g. 13, 24)
>> 30,82
   Missing value where no season is found between min and max dates)
   Give missing value code (e.g. -1)
>> 0
   Missing value for other cases (pixel not processed due to low amplitude,
   undefined class, failure to fit function etc.)
   Give missing value code (e.g. -2)
>> 1
   Give name to append to output files (no extension!) >> begin

   File type for the output image files
   2 = 16-bit signed integer (values will be rounded to nearest integer)
   3 = 32-bit real (no rounding)
   Specify file type (2 or 3)
>> 3

Opening begin_season1
Opening begin_season2
Opening begin_nseas
Opening begin_errors.txt
Opening begin_both_seasons

Working...
 Finished writing data
 File format 32-bit real
```

After the program has finished the run, view the resulting image with `TSM_imageview` (remember to use the same file format as the one you used when the file was created). The file `begin_nseas` contains the total number of seasons for each pixel. Over oceans it displays value 1 since this is the missing value code we specified, and over land it has found two seasons: This is logical since TIMESAT fits $n - 1$ seasons in unimodal areas, where n is the number of years in the data set. The file `begin_s1` shows the seasonality parameter for the first season identified within the time-interval specified. In the SE part of West Africa the season began around date 60 (i. e. $60 - 36 = 24$, meaning about day 240), whereas in the NW part of the area it began around date 69 (i.e. $69 - 36 = 33$, or about day 330). Note that some scattered pixels have value zero, indicating that no seasons was found within the specified time interval, and that the oceans have value 1, since this was the code that we assigned pixels not processed at all. The file `begin_s2` shows any secondary seasons found within the time period specified. Only a few scattered pixels are found, probably indicating the presence of noise at these locations. To investigate abnormal pixels we recommend using `TSM_GUI` to investigate the individual times-series at specific locations. Now, try generating some of the other parameters contained in the file `west_africa_TS.tpa`.

This is the end of the short tutorial. Output files generated are:

`seasonality.txt`	ASCII file containing seasonality data for a chosen pixel.
`west_africa.set`	ASCII files containing settings for running Timesat.
`west_africa_TS.tpa`	Binary file containing Timesat parameters.
`west_africa_fit.tts`	Binary file containing fitted data values.
`west_africa_raw.tts`	Binary file containing original time-series values.
`timeseriesfile.txt`	ASCII file containing fitted data values for selected pixels.
`west_africa_fit_0054`	32-bit real binary image file containing fitted data value for time step 54.
`begin_nseas`	Binary image file containing no. of seasons.
`begin_season1`	Binary image file containing the selected seasonality parameter for the first season identified within the time interval specified.
`begin_season2`	Binary image file containing the selected seasonality parameter for the second season identified within the time interval specified (if present).
`begin_both_seasons`	Binary image file containing seasonality parameters for both seasons within the time interval (if more than one is found).

9.9 Checklist for processing new vegetation index image data

In this section we provide a checklist that should help the user to define a settings file and process a new image data set of their own. Compare with the steps in the previous sections.

1. Create a new data directory where the vegetation index image files and, optionally, the quality image files should be saved.

2. Create a run directory where the output files from the Timesat processing should be written. Optionally the user may use the directory `timesat32\run`.

3. Check that the Matlab executables are on the Matlab path (cf. section 7.2).

4. Download or build vegetation index image files and (optional) images with quality data from some data provider, e.g. NASA. Save the files in the created data directory.

5. Determine and note the total number of images. Determine the number of years and the number of images per year.

6. Determine and note the number of rows and columns for the image files.

7. Convert all images into flat (header less) binary format. The file type should be one of the following: 8-bit unsigned integer, 16-bit signed integer, 32-bit real. For 16-bit signed integer please also determine if it is little or big endian byte order. The checks on the images are best done using `TSM_imageview` (cf. sections 9.3 and 10.2). File conversion can

be carried out with software such as the open source Geospatial Abstraction Data Base Library (GDAL), which can be downloaded from `www.gdal.org`.

8. Prepare an image file list. At the first row of the list the total number of vegetation index images should be given. After that comes the name and path of each of the images. If there are images missing generate a dummy image with values outside the normal data range. Write the name and path of the dummy image at the correct position in the image file list (cf. section 9.1). The user might want to duplicate the file names of the first and last years (cf. sections 4.1, 4.2, and 9.1). Give the image file list some suitable name and save it in the created directory along with all images. If quality data are available create a file list also for the quality images.

9. If available include a land cover map with the same spatial resolution and data format as the image files (cf. section 9.1). Use `TSM_imageview` to check the land cover map. Under *Color scale* select *qualitative* (cf. section 9.3).

10. Determine if there are areas with more than one vegetation season per year. Consider putting such an area in a separate land class.

11. Determine and note which part of the image you are going to process. `TSM_imageview` is helpful for this task.

12. Use `TSM_GUI` to explore your time-series data and fine tune the settings (cf. section 9.4). Pay attention to: spike method, fitting method, as well as number of envelope iterations, and adaptation strength. Explore the best method for determining the start and end of the season. If you have a land classes covering an area with two vegetation seasons per year the seasonal parameter should be put to a small value. If you have land classes covering an area with only one season per year the seasonal parameter should be put to 1. Carefully study sections 5.2 and 5.3 and make sure that you see the connection between the settings in the GUI and the items in the settings file.

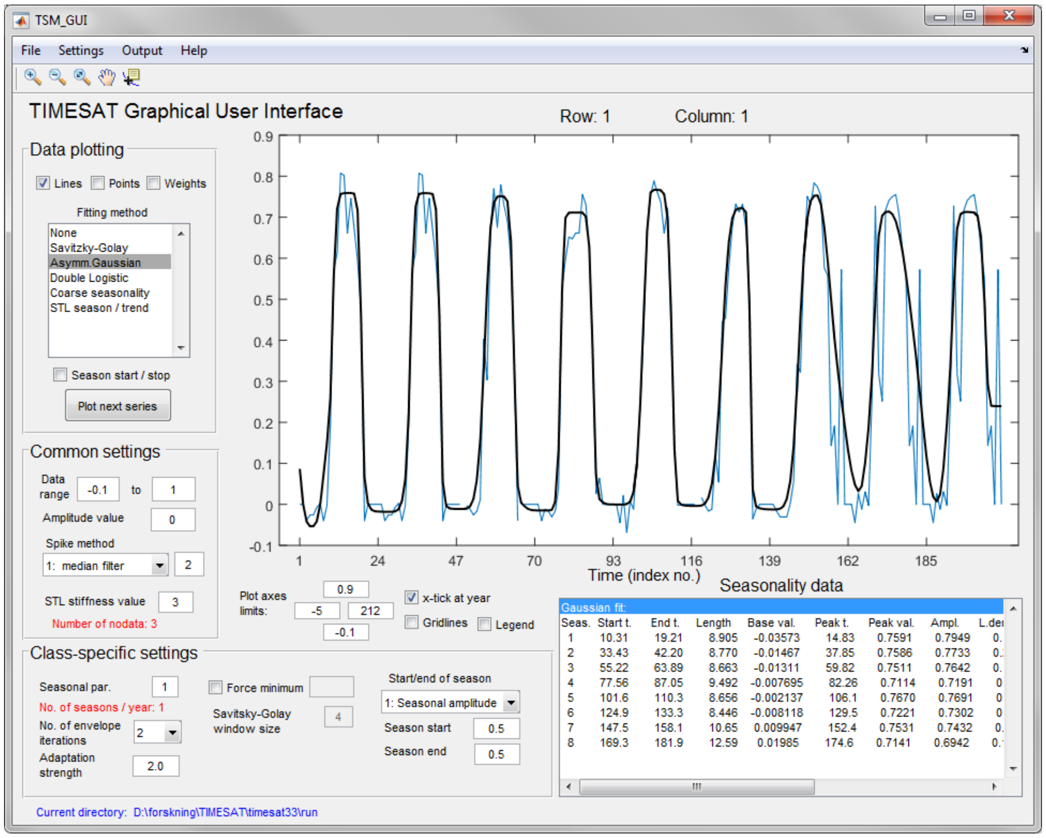
13. Use `TSM_settings` to generate the final settings file (cf. sections 5.2, 5.3, and 9.5). The settings file will be saved in the run directory.

14. Use `TSF_process` for processing the data. You may choose to run the program from the DOS command prompt or from `TSM_menu`.

15. We strongly advice the user to output and view fitted time-series (saved in a TIMESAT `.tts` file). Use `TSM_viewfits` to check the fits for as many pixels as possible. By spending time at this step the user can make sure that the fits turned out in the expected way. If things look strange then the settings must be tuned again.

16. Go to the run directory and check that the expected output files are in place. Do necessary post-processing to generate seasonality images etc (cf. section 9.8).

Part IV
Reference Manual



Row: 1 Column: 1

Lines ✓ Points Weights

Fitting method: Asymm.Gaussian

Season start / stop

Data range -0.1 to 1

Amplitude value 0

Spike method 1: median filter 2

STL stiffness value 3

Number of nodata: 3

Plot axes limits: 0.9, -5, 212, -0.1

✓ x-tick at year

Gridlines Legend

Seasonal par. 1

No. of seasons / year: 1

No. of envelope iterations 2

Adaptation strength 2.0

Force minimum

Savitsky-Golay window size 4

Start/end of season 1: Seasonal amplitude

Season start 0.5

Season end 0.5

Current directory: D:\forskning\TIMESAT\timesat33\run

Gaussian fit:

Seas.	Start t.	End t.	Length	Base val.	Peak t.	Peak val.	Ampl.	L.de
1	10.31	19.21	8.905	-0.03573	14.83	0.7591	0.7949	0.
2	33.43	42.20	8.770	-0.01467	37.85	0.7586	0.7733	0.
3	55.22	63.89	8.663	-0.01311	59.82	0.7511	0.7642	0.
4	77.56	87.05	9.492	-0.007695	82.26	0.7114	0.7191	0
5	101.6	110.3	8.656	-0.002137	106.1	0.7670	0.7691	0
6	124.9	133.3	8.446	-0.008118	129.5	0.7221	0.7302	0
7	147.5	158.1	10.65	0.009947	152.4	0.7531	0.7432	0.
8	169.3	181.9	12.59	0.01985	174.6	0.7141	0.6942	0.

10 Reference manual

This chapter contains specifications of all the programs included in TIMESAT. Programs coded in Matlab are given the prefix `TSM_` (TIMESAT Matlab). Programs coded in Fortran are all given the prefix `TSF_` (TIMESAT Fortran). The Fortran programs can be run either from the TIMESAT `TSM_menu` system or from the DOS prompt.

10.1 TSM_menu

`TSM_menu` is an interface for starting up all the different TIMESAT programs (see Figure 29). Matlab programs are started with the blue buttons and Fortran programs with the pink buttons. The buttons are organized from top to bottom in rough order of processing. The startup buttons are organized under *Data preparation*, *Data processing* and *Post-processing*. Fortran programs are run from a command window (DOS prompt) that opens in the current working directory. The name of the Fortran program folder will automatically be appended to the path of the command window.

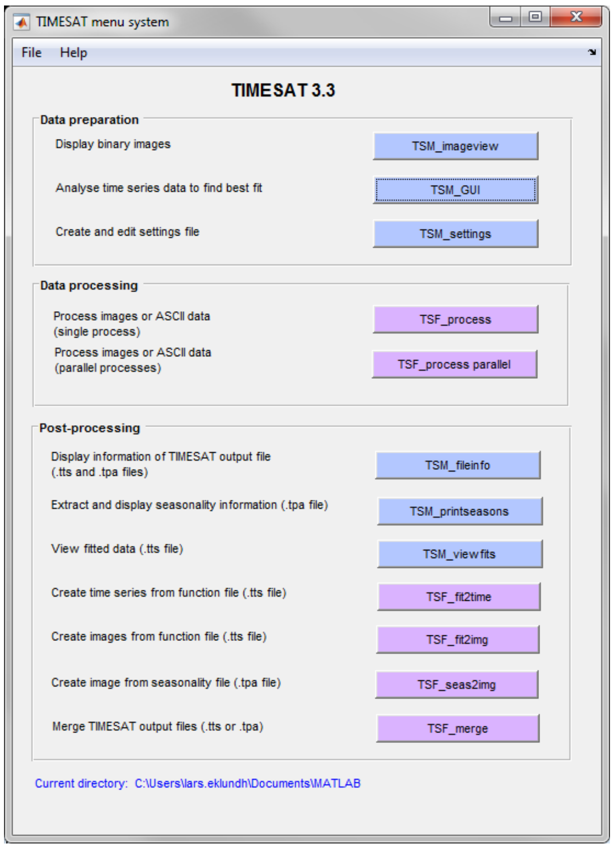

TIMESAT menu system
File
Help
TIMESAT 3.3
Data preparation
Display binary images
TSM_imageview
Analyse time series data to find best fit
TSM_GUI
Create and edit settings file
TSM_settings
Data processing
Process images or ASCII data (single process)
TSF_process
Process images or ASCII data (parallel processes)
TSF_process parallel
Post-processing
Display information of TIMESAT output file (.tts and .tpa files)
TSM_fileinfo
Extract and display seasonality information (.tpa file)
TSM_printseasons
View fitted data (.tts file)
TSM_viewfits
Create time series from function file (.tts file)
TSF_fit2time
Create images from function file (.tts file)
TSF_fit2img
Create image from seasonality file (.tpa file)
TSF_seas2img
Merge TIMESAT output files (.tts or .tpa)
TSF_merge
Current directory: C:\Users\lars.eklundh\Documents\MATLAB

Figure 29: *TIMESAT menu system. The system is divided into three logical areas: Data preparation, Data processing, and Post-processing.*

TIMESAT programs can be accessed also outside `TSM_menu` by typing their commands in the Matlab command window. Fortran commands are accessed from DOS command windows. If the command window has not been opened by TIMESAT it is necessary to add the directory with the Fortran executables to the DOS path before executing the commands. Alternatively give the command from the DOS prompt including the full path, e.g. to run `TSF_seas2img`:

```
c:\timesat33\timesat_fortran\tools\TSF_seas2img
```

The following menu choices are available in `TSM_menu`:

Top-level	**Sub-level**	**Contents**
File	Index TIMESAT file	Creates an index file for a `.tts` or `.tpa` file.
	Update old settings file	Updates versions settings files to version 3.3
	Preferences	Opens a dialogue for changing the current Matlab working directory
	Exit	Exits the menu system
Help	About	Displays version number of current TIMESAT release, and contact information to the authors.

10.2 TSM_imageview

This program (Figure 30) is particularly useful for quick display and for checking the formatting of flat (header less, unformatted) binary images. The user can vary settings for *Image file type*, *No. of columns* and *No. of rows* until the image displays correctly. Input files are binary images of the same kind as handled by the `TSF_process` program. These files are described in more detail under section 10.15. The program will issue a warning if the file size does not correspond with the specified image formatting.

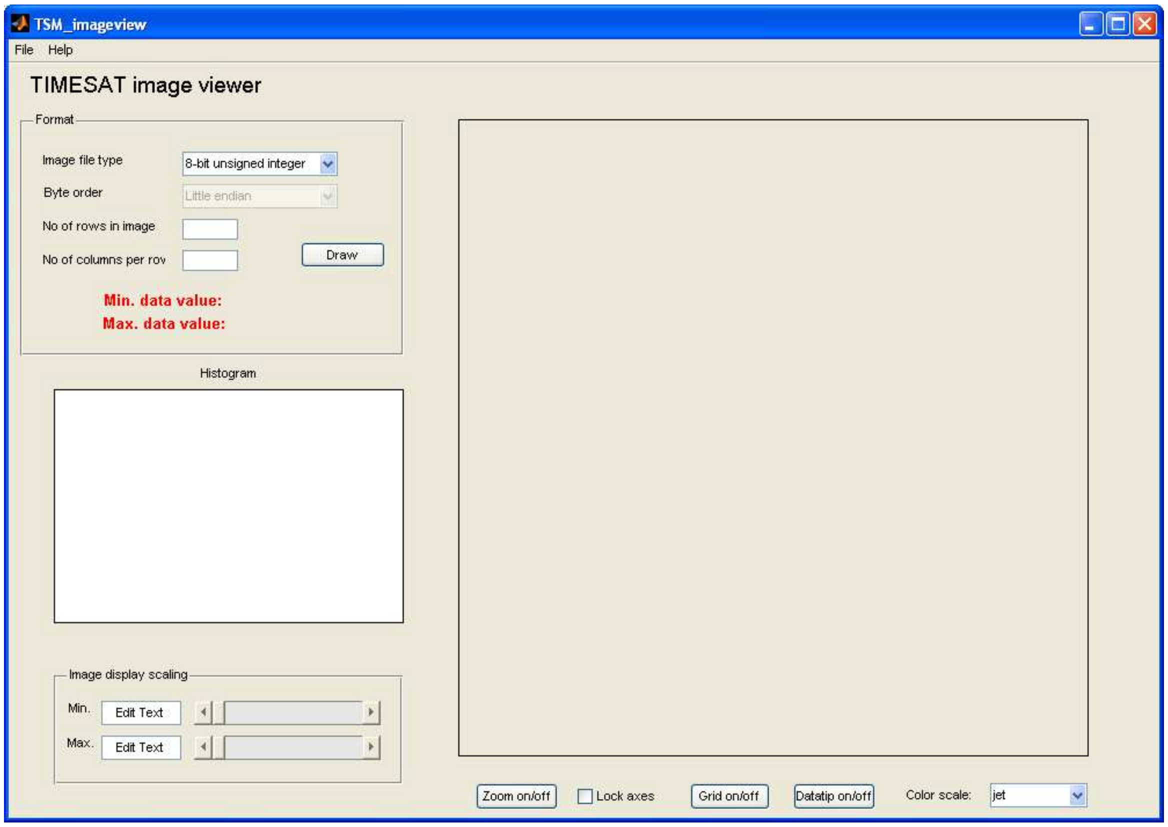


Image file type: 8-bit unsigned integer
Byte order: Little endian
No of rows in image
No of columns per row
Min. data value:
Max. data value:
Min.
Max.
Color scale: jet

Figure 30: *TIMESAT image view. This program is particularly useful for quick display and for checking the formatting of flat (header less, unformatted) binary images. The user can vary settings for* Image file type, No. of columns *and* No. of rows *until the image displays correctly.*

Some specific functions in the program are:

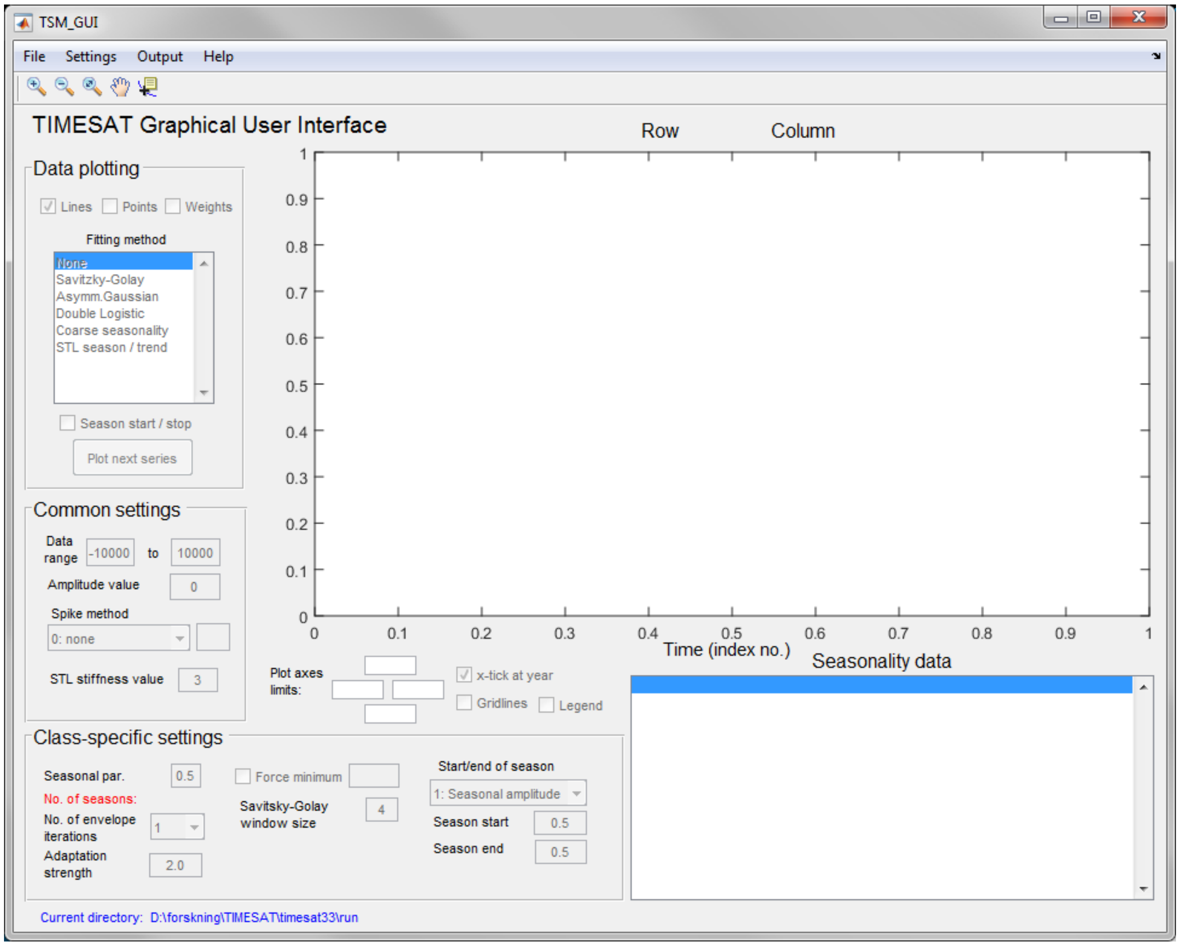
- identification of file types by testing different binary file type settings
- linear scaling of image data using the sliding bars or by specifying numeric values
- zooming using the Matlab default zooming functions
- locking the x- and y-axes. Useful when modifying the image scaling after zooming in
- display of pixel values by use of the Matlab datatip function, and choice of different color scales, including a scale for qualitative (nominal) data values, such as classified images

The following menu choices are available in `TSM_imageview`:

Top-level	**Sub-level**	**Contents**
File	Open image file	Opens a window for selecting an input image file
	Open file list	Opens a window for selecting an image file list. Click on a file in the list, then move over to `TSM_imageview` main window without closing the list, and fill out the choices under *Format* before pressing *Draw*
	Printing window	Opens a clone of the data plotting window with the default Matlab plotting tools, for printing and export to graphics files
	Exit	Exits menu system
Help	About	Displays an information text about `TSM_imageview`

10.3 TSM_GUI

This Matlab program (Figure 31) displays time-series data, fits functions to the data and outputs seasonality and fitted data for single time-series, from data located in binary vegetation index image files or ASCII time-series files. The program is useful for testing optimal settings for specific pixel locations, to be used for subsequent processing of full images using the Fortran program `TSF_process`. It is advisable to test a large number of pixels before deciding upon the settings for a specific image class. With `TSM_GUI` it is possible to go through profiles along columns or rows, or to specify square areas to process. The definition of settings is done by visually examining the function fits to the chosen time-series. The success of this process relates to the nature of the time-series, and the type of noise or disturbances in the data. Often a number of iterations will be necessary before a final fit has been achieved. This manual work is more of an art than a science. We have refrained from defining a goodness-of-fit statistic for the fits, since the best fits are often upper-envelope weighted, hence not unbiased, least-squares solutions. Note that the check boxes and options correspond to the settings described in sections 5.2 and 5.3. Some of the settings related to the image file type etc are set when the images or time-series are loaded (see section 10.4).



Lines [✓] Points [ ] Weights [ ]

Season start / stop [ ]

Data range: -10000 to 10000

Amplitude value: 0

Spike method: 0: none

STL stiffness value: 3

x-tick at year [✓] Gridlines [ ] Legend [ ]

Seasonal par.: 0.5

No. of seasons:

No. of envelope iterations: 1

Adaptation strength: 2.0

Force minimum [ ]

Savitsky-Golay window size: 4

Start/end of season: 1: Seasonal amplitude

Season start: 0.5

Season end: 0.5

Current directory: D:\forskning\TIMESAT\timesat33\run

Figure 31: *TSM_GUI. This is the main program for fitting functions to time-series data for selected pixel locations or for single ASCII data files.*

The `TSM_GUI` program contains, in addition to all the check boxes and other options, the following menus:

Top-level	Sub-level	Contents
File	Open list of image file	Opens a window for defining input from binary files
	Open ASCII data file	Opens a single ASCII file containing time-series data
	Printing window	Opens a clone of the data plotting window with the default Matlab plotting tools, for printing and export to graphics files
	Exit	Exits `TSM_GUI`
Settings	Load settings file	Loads a predefined settings file
	Save to settings file	Opens the `TSM_settings` tool for saving the current `TSM_GUI` settings to a file `*.set`
Output	Write seasonality data to file	Writes seasonality data to the file `seasonality.txt` computed from fitted functions using the current settings
	Write fitted function to file	Writes fitted functions to files `sg.txt`, `logistic.txt`, or `gauss.txt` depending on the choice
Help	About	Displays a short information text

10.4 Data input for TSM_GUI

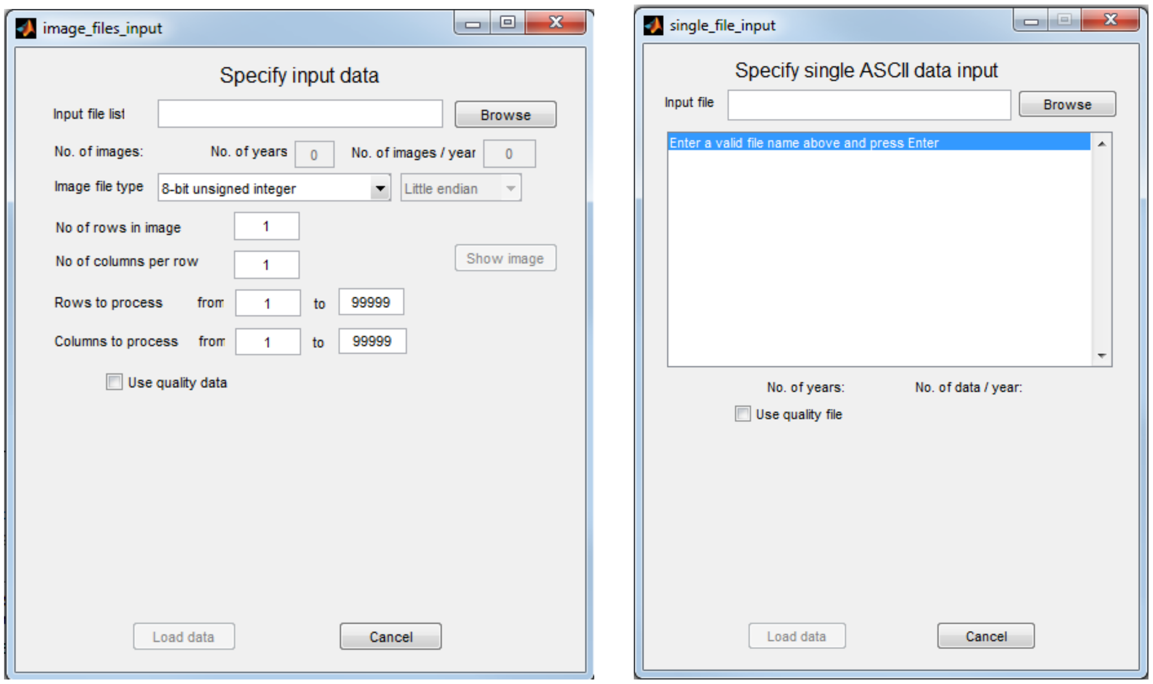
Loading binary image data

When selecting *File, Open list of image files* in `TSM_GUI` the window `image_files_input` (Figure 32) is loaded. Specify a file containing a list of binary input images (see section 9.1) under *File lists*. The file names should be given with a relative or full path. The format of the image files listed is specified in section 10.15 under *Image data files*.
After specifying the input file list, the data format is given in the window. Note that *No. of images/year* will be automatically computed from the data in the file and the value given under *No. of years*, *No of rows in image*, and *No of columns per row* should define the size of each input file. However, *Rows to process* and *Columns to process* define the specific area that `TSM_GUI` will attempt to load. `TSM_GUI` loads all specified data into primary memory, hence, defining a very large area may lead to memory allocation problems. It is thus advisable to process only a small portion of the data at the time. Use the button *Show image* to get a preview of the binary images. Note that loading of large images can be very slow. In preview, a box defining the rows and columns to be processed can easily be defined graphically.

Select *Use quality* data when quality data for weighting the image data are to be used. The quality images should have formatting that is identical to the input images. Three data intervals and three corresponding weights associated with each interval are specified. The *Load data* button will not be activated until all necessary values have been provided.

Loading data from ASCII file

When selecting *File, Open ASCII data file* in `TSM_GUI` the window `single_file_input` will be loaded (Figure 32). The name of the input file is given, and a preview of the file will be loaded into the window. The format of the file is defined in sections 2.1 and 10.15. If quality


image_files_input

Specify input data

Input file list | Browse

No. of images: No. of years 0 No. of images / year 0

Image file type 8-bit unsigned integer | Little endian

No of rows in image 1

No of columns per row 1 | Show image

Rows to process from 1 to 99999

Columns to process from 1 to 99999

☐ Use quality data

Load data | Cancel

single_file_input

Specify single ASCII data input

Input file | Browse

Enter a valid file name above and press Enter

No. of years: No. of data / year:

☐ Use quality file

Load data | Cancel

Figure 32: *Window for loading binary images (left) and data from ASCII files (right).*

data are available, these can be loaded by specifying *Use quality file*. Quality data should be stored in a file of identical formatting as the sensor data. To activate the *Load data* button, both data file and quality file need to be specified.

10.5 Settings in TSM_GUI

Settings in `TSM_GUI` will be implemented immediately for the time-series. Display can be slow when multiple processing is done, e.g. fitting with different methods, or fitting very long data series. The choices under *Data plotting* control the display of the current time-series. When selecting *Savitzky-Golay*, *Asymm. Gaussian*, *Double Logistic*, or *STL season / trend*, the fitted data will be displayed along with the original data, and seasonality parameters will be shown in the window *Seasonality data*. Choosing Coarse seasonality displays the results of the sinusoidal fit (section 3.5). The button *Plot next series* will load the next data series (column and row when input is from images, or next line when input is from ASCII file). Under *Common settings* settings that will be valid for all pixel classes are found. These typically define criteria for including or excluding time-series, e.g. if the amplitude is too low, or if too few data values lie within the data range. Under *Class-specific settings* those settings are defined that can be applied differently to different classes when running the `TSM_process` module. `TSM_GUI`, however, does not recognize different classes when processing time-series. The meaning of the different settings is defined in section 5.3 in the description of the settings file.

When the user is satisfied with the settings for a given area or land class, these can be stored to file by going to *Settings*, *Save to settings file*. This starts the `TSM_settings` program and transfers the current settings to that program. Values can be modified, classes can be added, and data can be stored to a settings file. Values modified in `TSM_settings` will not be returned to `TSM_GUI`.

10.6 Output files from TSM_GUI

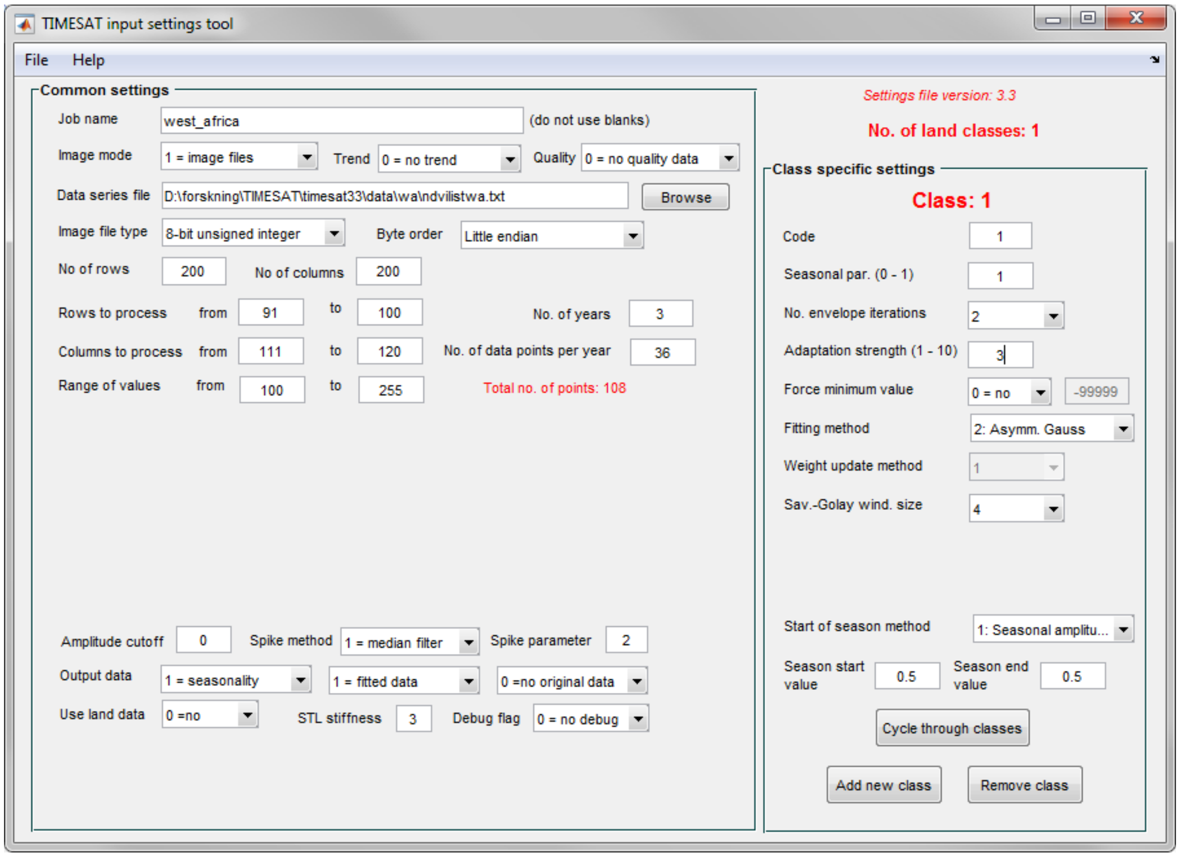
Various ASCII data can be written from `TSM_GUI` for selected time-series. These will be written to the working directory under the following names:

`seasonality.txt`	Seasonality data for each season
`sg.txt`	A single line of fitted values using the Savitzky-Golay method
`gauss.txt`	A single line of fitted values using the asymmetric Gaussian method
`logistic.txt`	A single line of fitted values using the double logistic method
`STL_season.txt`	A single line representing the STL seasonal curve
`STL_trend.txt`	A single line representing the STL trend line.

10.7 TSM_settings

`TSM_settings` is a tool for creating and managing TIMESAT settings-files (Figure 33). It can be called from the Matlab command line, from `TSM_menu` or from `TSM_GUI`. Settings files have the extension `.set` and are used for storing settings that will affect the way `TSM_process` processes data into seasonality or smoothed data. A settings file is an ASCII file that can be edited directly with any file editor, or managed with the `TSM_settings` program. Detailed explanations of each of the parameters are given in section 5.3.

The main window is divided into two sections. Under *Common settings* general settings, e.g. input files, processing window, amplitude cutoff, spike filtering method, and mode of analysis are defined. Also the name of optional quality files and a class file are defined here. Under *Class specific settings* each category of settings for specific classes are defined. If only one class is used (the default if no land class file is specified), the settings defined for class 1 will be used for all data processing. Up to 255 classes can be defined. Class settings can be viewed, added or removed by the buttons *Cycle through classes, Add new class, and Remove class.*



Common settings

Job name: west_africa (do not use blanks)

Image mode: 1 = image files; Trend: 0 = no trend; Quality: 0 = no quality data

Data series file: D:\forskning\TIMESAT\timesat33\data\wa\ndvilistwa.txt

Image file type: 8-bit unsigned integer; Byte order: Little endian

No of rows: 200; No of columns: 200

Rows to process from 91 to 100; No. of years: 3

Columns to process from 111 to 120; No. of data points per year: 36

Range of values from 100 to 255; Total no. of points: 108

Amplitude cutoff: 0; Spike method: 1 = median filter; Spike parameter: 2

Output data: 1 = seasonality; 1 = fitted data; 0 =no original data

Use land data: 0 =no; STL stiffness: 3; Debug flag: 0 = no debug

Settings file version: 3.3

No. of land classes: 1

Class specific settings

Class: 1

Code: 1

Seasonal par. (0 - 1): 1

No. envelope iterations: 2

Adaptation strength (1 - 10): 3

Force minimum value: 0 = no; -99999

Fitting method: 2: Asymm. Gauss

Weight update method: 1

Sav.-Golay wind. size: 4

Start of season method: 1: Seasonal amplitu...

Season start value: 0.5; Season end value: 0.5

Figure 33: *TSM_settings window. Common settings are defined in the left pane and class specific in the right pane.*

`TSM_settings` contain the following menus

Top-level	Sub-level	Contents
File	Open settings file	Loads settings from a `.set` file. This choice is not activated when starting the program from `TSM_GUI`
	Update old settings file	Updates older version settings files to version 3.3
	Save settings file as	Saves settings to a `.set` file
	Add current class settings to file	Adds settings for the currently visible class to the end of an existing settings file. This choice is useful when of defining settings for different classes. An initial `.set` file is created for the first class. When subsequent class settings have been defined in `TSM_GUI`, `TSM_settings` is loaded and the current settings appended to the file using this option.
	Exit	Exits `TSM_settings`
Help	About	Displays an information text about the program

10.8 TSF_process

This is the core module in the TIMESAT package. It is used for processing full images or collections of ASCII time-series and to generate TIMESAT seasonality and/or smoothed data files. `TSF_process` is started from `TSM_menu` or from the command line in a DOS window. When executing the program the user will be prompted to give the name of an existing settings file. After the execution, `TSF_process` will write the names of the output files to the command window. The algorithms and the functionality of the programs are discussed in Part II Algorithm Theoretical Basis, and the output files are described in sections 6.1, 6.2, and 10.16. In version 3.1 the program `TSM_process` was also accessible from the TIMESAT menu. From version 3.2 it has been removed from the menu but, for compatibility reasons, remains as a program that can be executed from the Matlab command line.

Question	Explanation
Give name of input file	Specify settings file
Give number of processors	Specify 1 for running directly, 2 or higher for starting a parallel processing job (see section 10.9)

Command line

`TSF_process` *settings_file no._ of_ processors*

10.9 TSF_process parallel

Using this option enables execution of `TSF_process` in parallel mode. It is a two-step process. In the first step, the user specifies the number of processes to run. This is determined by the available processors on the current machine. If e.g. 10 is specified, the job will be divided into ten equal parts, and ten individual settings files will be generated. In Linux a report of this step is written to a file called output which can be viewed to make sure that it has completed correctly. A script file named `jobname_script.bat` (Windows) or `jobname_script.sh` (Linux) will be generated. The script file contains commands for starting the parallel jobs, for merging the resulting files (see `TSF_merge` below), and for deleting temporary files. In the second step the script file is executed. This is done by typing `jobname_script` (DOS) or `./jobname_script.sh` (Linux) at the command prompt. When the jobs are executed reports will be written to files called `output1.txt`, `output2.txt` ...etc. (Windows) or `output1`, `output2`... etc. (Linux). These can be viewed with an editor to see the status of the jobs, and to ensure that the jobs have finished without errors. Please note that a number of operating system commands are used in the script files which are not always identical on different operating system versions. The Linux version uses the Unix Bash shell; users running other shells may be required to rewrite the script. To ensure correct execution of the jobs the user is advised to first try out parallel processing on a small subset of the data to make sure that the whole process is completed without errors. It should also be noted that it is not advisable to use the maximum number of processors on the machine (or exceed it) since this may deplete all system resources.

10.10 TSF_readheader (obsolete)

This program is obsolete but kept in the Fortran folder for backwards compatibility. It is replaced by `TSM_fileinfo`.

10.11 TSM_fileinfo

This program is used for displaying the structure of a TIMESAT output file (`.tpa` or `.tts`). The output information will display the following data: no. of years, no. of points per year, total no. of points, no. of rows and columns, extent of processing window. In addition it will display a map showing the number of seasons (`.tpa`) or number of years (`.tts`) of data stored in the file. The map is to help the user to identify the geographical area that has been processed. If single time-series data are processed the map is not displayed.

10.12 TSM_printseasons

This program is meant for viewing seasonality parameters stored in a TIMESAT `.tpa` file. When started from `TSM_menu` the program will prompt you to respond to the questions below

Question	**Explanation**
Give name of input fitted data file	Name of input file (only when started from the command prompt)
First row:	First row in processing window
Last row:	Last row in processing window
First column:	First column in processing window
Last column:	Last column in processing window
Name of output text file (hit Enter to print to screen):	Name of the output file

Seasonality parameters will be shown for the first pixel window. By hitting *Enter* you can view parameters for the remaining pixels in the window. To abort the program press *ctrl-c*.

10.13 TSM_viewfits

This program is meant for viewing data stored TIMESAT `.tts` files. The program will prompt you to respond to the questions below

Question	**Explanation**
Give name of input data files	Name of input `.tts` files (`raw` and/or `fit`).
First row:	First row in processing window
Last row:	Last row in processing window
First column:	First column in processing window
Last column:	Last column in processing window

A plot window in Matlab will open with fitted data for the first pixel. By hitting *Enter* you can view the remaining pixels in the window. You will need to close the plotting window manually. To abort the program press *ctrl-c*.

10.14 TSF_fit2time

This program extracts time-series data (raw or fitted) for one or many pixels and writes to an ASCII file. When starting from `TSM_menu` you will be asked to choose a `.tts` file containing the data. A number of questions will follow. Extracted data are written row by row (see sections 9.8 and 2.1).

Question	**Explanation**
Name of input fitted data file (*infile*)	Name of input file (only when started from the command prompt)
Specify image window to extract data for (*frow, lrow, fcol, lcol*)	Provide first row, last row, first column, last column of window to be processed
Name of output file: (*outfile*)	Name of ASCII file containing the resulting data. This file will contain a header with number of years, number of points per year, and number of time-series. Data values for pixels are written row by row.

Command line

`TSF_fit2time` *infile frow lrow fcol lcol outfile*

10.15 TSF_fit2img

Creates one or several vegetation index images from the fitted data by the fitting methods Gaussian, Logistic or Savitzky-Golay. When starting from `TSM_menu` you will be asked to choose a `.tts` file containing fitted data. A dialogue appears:

Question	**Explanation**
Name of input fitted data file (*infile*)	Name of input file (only when started from the command prompt)
Code to give to missing pixels (*misspix*)	Provide a numerical code to replace missing data with in output files. This code should be outside the range of the input data
Name of output file (no extension): (*outfile*)	Name of output binary image file(s), without file extension. Do not use spaces.
Specify the file type for the output image files (1,2 or 3) 1 = 8-bit binary 2 = 16-bit signed integer 3 = 32-bit real	Give an integer code 1-3 specifying the output data. This should correspond to the data type of the original input data.
Image no. to map (−1 = all)	Sequence number of the time for which the fitted data should be mapped. Give −1 to map all data.

Command line

`TSF_fit2img` *infile misspix outfile filetype image no.*

The program will run and generate binary image files containing the fitted data values. A sequence number will be appended to the name specified in the dialogue. Image files can be viewed with `TSM_imageview`.

10.16 TSF_seas2img

Program for creating an image from the seasonality data generated by TIMESAT. Starts by asking for the file of a seasonality `.tpa` file and this is followed by dialogue below:

Question	**Explanation**
Name of input seasonality file (*infile*)	Name of input file (only when started from the command prompt)
Seasonal parameter to output (*seaspar*)	Provide a numerical code for the desired parameter to output according to the list given
Specify dates between which the season is expected to occur (*datemin datemax*)	Give two sequence numbers defining the start and end dates between which the seasonal maximum is found. (In previous TIMESAT versions the full season had to be found within these dates.) To ensure that the full season is captured the interval can be made somewhat wider than the expected season. For detailed explanation see Part II Algorithm Theoretical Basis section 6.3
Code to put if season is not found between min and max dates. (*missseason*)	Give an integer missing data code to assign pixels where no seasonal maximum was found between the dates specified
Code for missing pixel for other reason (*misspix*)	Give an integer missing data code representing missing data for other reasons than the above, e.g. due to failure to fit one of the functions due to too many missing input data values for that pixel
Name to append to output files (*outfile* no extension!)	Name of output files (no spaces, no file
Specify the file type for the output image files (2 or 3) 2 = 16-bit signed integer (values will be rounded to nearest integer) 3 = 32-bit real (no rounding)	Format of binary output files

Command line

`TSF_seas2img` *infile seaspar datemin datemax misseason misspix nameout filetype*

The following output files will be generated:

`name_s1`	Binary file of data for the first season found within the interval
`name_s2`	Binary file of data for the second season found within the interval (if dual seasons exist)
`name_nseas`	Binary file containing the number of seasons
`name_errors.txt`	ASCII file of errors. Usually empty.

10.17 TSF_merge

Merges `.tts` or `.tpa` files for adjacent areas. Useful if `TSF_process` has been run for smaller windows that need to be merged. Files must be adjacent and can be merged by row (N-S) or by column (E-W). The order of input files matters and must be from top down (merging by row), or from left to right (merging by column). Dialogue

Question	**Explanation**
Merge by columns (0) or by rows (1)	Mode of merging
Give number of input data files to be merged	No. of input files
Give name of file 1	Name of `.tts` or `.tpa` file
Name of output file	Name of merged `.tts` or `.tpa` file
Command line	
`TSF_merge` *rowmerge nfiles infile1 infile2 ...outfile*	

10.18 Running from the command prompt to automate processing

`TSF_process` and the other Fortran programs can be executed without first starting Matlab. It is then necessary to reference the full program including the path (e.g. `c:\TIMESAT33\timesat_fortran\main\TSF_process`), or to add the program folders to the DOS or Linux path. An alternative is to copy the executable files into the run directories and execute from there. In Windows the programs are named `*.exe` and in Linux they are named `*.x64` (e.g. `TSF_process.x64`).

It is possible to automate processing by starting jobs from script files that can execute a series of TIMESAT commands, e.g.:

```
TSF_process sahel.set
TSF_fit2img sahel_fit.tts 0 sahel_image.rst 3 -1
TSF_seas2img sahel_TS.tpa 3 33 75 0 0 length 3
```

In Windows the script file must be named `*.bat` and is executed by typing the name at the command prompt. In Linux any name could be used.

10.19 Working in Linux as compared to Windows

Most TIMESAT functions in the Linux version are identical to those in Windows (as described in this manual), however certain differences exist. Notably, all file directories are specified with a forward slash (`/`) instead of the backward slash. Furthermore, execution of the Fortran programs differs somewhat between Windows and Linux. First of all, it may be necessary to modify the file rights to allow execution of the Fortran executables in `timesat_fortran/main` and `timesat_fortran/tools`. This can be done using the command: `chmod 755 *.x64`. A further modification that may be necessary if processing a large number of files is to execute the following line (or add to the `.bashrc file`): `ulimit -n 2400` (allows opening of up to 2400 files simultaneously).

When executing a Fortran command from TIMESAT, no command window is opened, but the commands are executed under the Matlab prompt using the `!` (bang) command. Also other system commands will have to be executed from the Matlab prompt in the same way (e.g. `!more output` to monitor the progress of `TSF_process`). Alternatively a shell window can be opened and the Fortran programs executed from there. In that case the path to the Fortran executables must be appended to the Linux path:

```
PATH=/myhome/timesat33/timesat_fortran/main:/myhome/timesat33/timesat_fortran/tools:$PATH
```

This is necessary when running parallel jobs in the precompiled TIMESAT version, since no command line is directly accessible from TIMESAT. Execution of a script residing in the current directory is done with the command: `./script_file`. The script file generated when processing in parallel is written with bash shell commands. Other shells may require modification of this file.

10.20 Input files for TIMESAT

Vegetation index image files

Image data to be processed in TIMESAT, e.g. data from a satellite sensor, should be formatted as headerless (flat) binary files, organized as sequential data streams by column and row. Values can be stored as 8-bit integer (0 – 255), 16-bit signed integer (-32767 to $+32768$), or 32-bit real (decimal values between 1.2×10^{-38} and 3.4×10^{38}). One file is used for storing a single image. This is a generic format that can be imported/exported by most image processing software. The user has to keep track of the organization into columns and rows. TIMESAT is image oriented and does not consider geographical coordinate systems other then column and row. Column and row numbers begin with 1.

Quality image files

These files contain numeric values corresponding to the observation quality of the image data. The file format and organization should be identical to that of the the vegetation index image data. The quality values will be translated into a maximum of three classes. Numeric values that will belong to a certain weight class need to be grouped, e.g. 1 – 10 corresponding to the weight 0.1, 11 – 20 corresponding to the weight 0.5, and 21 to 30 corresponding to the weight 1.0. The assignment of weights, e.g. 0.1, 0.5 and 1.0 is defined in a settings file and determine the weight of the data point in the least-squares fit. A weight of zero will mean that the observation is excluded from the fit.

Image file lists

ASCII files containing the names of all binary input files. One separate file list should exist for all vegetation index image files and one for all quality image files. The format is as follows:

```
N
path\imagename_1
path\imagename_2
 ....
path\imagename_N
```

where N is the number of image files. File names of image should be in chronological order, and include the complete path to the file, unless the files are located in the working directory from which TIMESAT is run. Only the N first images in the file will be processed.

ASCII data file

This is a file containing data organized as separate time-series for one or more locations. The file should be formatted as follows:

$$\begin{array}{l} nyear\ nptperyear\ nts \\ \left.\begin{array}{llll} y_1 & y_2 & \cdots & y_N \\ y_1 & y_2 & \cdots & y_N \\ & & & \\ y_1 & y_2 & \cdots & y_N \end{array}\right\} nts \end{array}$$

where *nyears* gives the number of years spanned by the time-series, *nptperyear* is the number of data values in one year, and *nts* is the number of data points. All time-series in the file need to have identical numbers of years and data values per year.

Land cover file

This is a binary image file, organized like the binary vegetation index image files, and with identical geometry and file format. Land cover codes should be integer numbers 0 – 255. If stored as real numbers these will be rounded to the nearest integers.

Settings file

This is an ASCII file containing settings for running `TSF_process` or `TSM_process`. It should reside in the run directory. The file is created and modified using `TSF_settings`, but can also be edited with an ASCII file editor. The file contains information and one comment per row. The table below shows the layout of the file (see also section 5.3 for a detailed description of the settings).

Row	Example	Short description	Explanation
1	`Version: 3.3`		Keeps track of the settings file version
2	`west_africa`	Job name	Job name (no blanks) - max 100 chars. This will determine the name of output files from TIMESAT
3	1	Image/series mode (1/0)	1 = image mode, 0 = ASCII time-series
4	0	Trend (1/0)	1 = STL trend fitting activated. Overrules choice of fitting method (row 32).
5	1	Use quality data (1/0)	1 = use quality data, 0 = do not use quality data
6	`datalist.txt`	Data file list/name	Name, followed by `%`, of file list (for images) or data file name (for ASCII data).
7	`quallist.txt`	Quality file list/name	Name, followed by `%`, of quality list (for images) or quality file name (for ASCII data)
8	1	Image file type	1 = 8-bit unsigned integer, 2 = 16-bit signed integer, 3 = 32-bit real
9	0	Byte order (1/0)	0 = little endian byte order, 1 = big endian byte order (for 16-bit integers)

10	200 200	Image dimension	No. of rows in image, and no. of columns per row
11	111 120 91 100	Processing window	Window to process (start row, end row, start column, end column)
12	3 36	Years and points per year	No. of years and no. of points per year
13	1 255	Valid data range	Lower and upper data values for valid range. Data outside range will be assigned weight 0
14	1 12 0.1	Quality range 1 and weight	Lower and upper values for quality class 1 and assigned weight
15	13 22 0.5	Quality range 2 and weight	Lower and upper values for quality class 2 and assigned weight
16	23 31 1	Quality range 3 and weight	Lower and upper values for quality class 3 and assigned weight
17	0	Amplitude cutoff value	Cutoff for low amplitude. Series with amplitude smaller than this value will not be processed. 0 processes all data
18	0	Debug (3/2/1/0)	Debug flag. 1 - 3 = print debug data, 0 = do not print debug data
19	1 1 0	Output files (1/0 1/0 1/0)	Flags for output data (seasonality, fitted data, and original data)
20	0	Use land cover (1/0)	1 = use land cover map, 0 = do not use land cover map
21	landcoverdata	Name of land cover file	Name, followed by %, of land cover file
22	1	Spike method (3/2/1/0)	Spike method. 3 = weights from STL multiplied with original weights, 2 = weights from STL, 1 = method based on median filtering, 0 = no spike filtering
23	2	Spike value	If spike method = 1 the spike value determines the degree of spike removal. A low value will remove more spikes
24	0	STL stiffness value	Parameter for STL trend stiffness. Varies between 1.0 and 10.0; default = 3.0.
25	2	No. of land cover classes	No. of land cover classes (if land cover data are used)
26	************	Separator	After separator comes class specific parameters
27	1	Land cover code for class 1	Land cover code for class 1
28	1	Seasonality parameter (0-1)	A value near 1 will attempt to fit one season per year, a value close to zero will attempt to fit two seasons

29	3	No. of envelope iterations (3/2/1)	No. of iterations for upper envelope adaptation (3,2,1). Choosing 1 means no envelope adaptation.
30	2	Adaptation strength (1-10)	Strength of the envelope adaptation. 10 is the maximum strength
31	0 0	Force to minimum (1/0) and value of minimum	Force to minimum. 1 = points below given minimum value will be forced to the specified minimum value. 0 = no forcing to value
32	3	Fitting method (3/2/1)	Fitting method. 3 = double logistic, 2 = asymmetric Gauss, 1 = Savitzky-Golay If STL trend fitting activated (row 4) this overrides the fitting method
33	1	Weight update method	Weight update method (not in use)
34	4	Window for Savitzky-Golay	Half window for Savitzky-Golay filtering. A large value of the window will give a high degree of smoothing
35	0	Reserved	Not in use
36	0	Reserved	Not in use
37	1	Season start start/end method (4/3/2/1)	Method for determining start/end of season based on intersection of the fitted curve. 4 = STL trend: at the intersection with the trend line from STL. 3 = Relative amplitude: at the point where the curve intersects a proportion of the relative seasonal amplitude. 2 = Absolute value: at the point where the curve intersects an absolute value in units of the data. 1 = at the point where the curve intersects a proportion of the seasonal amplitude.
38	0.5 0.5	Season start / end values	Values for determining season start/end If start method is 1 or 3 the values must be between 0 and 1
39 – 51		Separator and data for class 2	Same as rows 26–38, but for class 2
52 – 64		Separator and data for class 3	Same as rows 26–38, but for class 3
· · ·		· · ·	· · · etc. for a maximum of 255 classes

10.21 Output files for TIMESAT

`TSF_process` and `TSM_process` generate three types of data: (1) seasonality data, written to the file `jobname_TS.tpa`, (2) fitted functions, written to the file `jobname_fit.tts`, and (3) original data, written to the file `jobname_raw.tts`. The jobname is supplied in the settings file and can be any name with maximum 100 ASCII characters, but no spaces.

File with seasonality data

The `.tpa` files contain extracted TIMESAT seasonality parameters and have the following structure:

$$\begin{array}{l}
nyears \quad nptperyear \quad rowstart \quad rowstop \quad colstart \quad colstop \\
row_1 \quad col_1 \quad n_1 \\
\left.\begin{array}{ccccccccccccc}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
 & & & & & \vdots & & & & & & & \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13}
\end{array}\right\} \begin{array}{l} n_1 \text{ lines with values} \\ \quad \text{one line per season} \end{array} \\
row_2 \quad col_2 \quad n_2 \\
\left.\begin{array}{ccccccccccccc}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
 & & & & & \vdots & & & & & & & \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13}
\end{array}\right\} \begin{array}{l} n_2 \text{ lines with values} \\ \quad \text{one line per season} \end{array} \\
\qquad \vdots \\
row_M \quad col_M \quad n_M \\
\left.\begin{array}{ccccccccccccc}
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} \\
 & & & & & \vdots & & & & & & & \\
p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13}
\end{array}\right\} \begin{array}{l} n_M \text{ lines with values} \\ \quad \text{one line per season} \end{array}
\end{array}$$

Here $n_1, n_2, \ldots, n_M$ give the number of full seasons for which seasonality information have been determined. This information is followed by seasonality parameters $p_1, p_2, ..., p_{13}$ (see section 4.4) for each of the the full seasons. The variables *row*, *col*, and n are 32-bit integers whereas seasonality parameters $p_1, p_2, ..., p_{13}$ are written as 32-bit single precision real numbers.

Files with time-series

The `.tts` files contain fitted data and original data and have the following structure:

$$\begin{array}{l}
nyears \quad nptperyear \quad rowstart \quad rowstop \quad colstart \quad colstop \\
row_1 \quad col_1 \\
y_1 \quad y_2 \quad y_3 \quad \cdots \quad y_{N-1} \quad y_N \\
row_2 \quad col_2 \\
y_1 \quad y_2 \quad y_3 \quad \cdots \quad y_{N-1} \quad y_N \\
\qquad \vdots \\
row_M \quad col_M \\
y_1 \quad y_2 \quad y_3 \quad \cdots \quad y_{N-1} \quad y_N
\end{array}$$

where *nyears* gives the number of years spanned by the time-series, *nptperyear* is the number of data values in one *rowstart*, *rowstop*, *colstart* and *colstop* are variables defining the processing window, row_1, $row_2, \ldots, row_M$ are row numbers, $col_1, col_2, \ldots, col_M$ are column numbers, and y_1, $y_2, \ldots, y_N$ are the fitted or original data values. If input is from an ASCII file, columns will always be set to one. The variables *nyears*, *nptperyear*, *rowstart*, *rowstop*, *colstart*, *colstop*, $row_1, \ldots, row_M$, $col_1, \ldots, col_M$ are 32-bit integers, whereas $y_1, \ldots, y_N$ are 32-bit real data. Apart from the output files generated in `TSF_process` and `TSM_process`, ASCII files containing information for single file locations are generated in `TSM_GUI`. Files generated in the post-processing programs `TSF_fit2time`, `TSF_fit2img` and `TSF_seas2img` are described under each of these chapters.

10.22 Index files

Timesat generates index files that allow for faster access of the output data files. The output files can be very large, and the index files can considerably speed up access to specific locations. This is particularly noticed when plotting data using e.g. the routines `TSM_printseasons` and `TSM_viewfits`. The index files have the extension `.ndx`, and are automatically generated when running the post-processing programs mentioned above, if required. They can also be generated from the `TSM_menu` under File. The format is described in chapter 6.5.

11 Acknowledgements

Support from the Swedish National Space Board, the Crafoord foundation, and the Swedish Research Council FORMAS is gratefully acknowledged. The authors are thankful for all encouragement, comments and suggestions for improvements from the Timesat user community. We would especially want to thank Hua Yuan, Beijing Normal University, Claudia de Pus, Gent, and colleagues at Lund University: Jonas Ardö, Zhanzhang Cai, Roger Groth, Margareta Hellström and Hongxiao Jin.

12 References

Barichivich, J., Briffa, K.R., Myneni, R.B., Osborn, T.J., Melvin, T.M., Ciais, P., Piao, S. and Tucker, C. (2013), Large-scale variations in the vegetation growing season and annual cycle of atmospheric CO2 at high northern latitudes from 1950 to 2011, Global Change Biology, 19, 3167-3183.

Beck, P.S.A., Jönsson, P., Högda, K.-A., Karlsen, S. R., Eklundh, L. and Skidmore, A.K., 2007, A ground-validated NDVI dataset for monitoring vegetation dynamics and mapping phenology in Fennoscandia and the Kola peninsula. International Journal of Remote Sensing, 28, 4311-4330.

Cleveland, R.B., Cleveland,W.S., McRae, J.E., and Terpenning, I., 1990, STL: A Seasonal-Trend Decomposition Procedure Based on Loess. Journal of Official Statistics, 6, 3-73.

Eklundh, L., Johansson, T. and Solberg, S., 2009, Mapping insect defoliation in Scots pine with MODIS time-series data. Remote Sensing of Environment, 113, 1566-1573.

Eklundh, L. and Jönsson, P., 2003, Extracting Information about Vegetation Seasons in Africa from Pathfinder AVHRR NDVI Imagery using Temporal Filtering and Least-Squares Fits to Asymmetric Gaussian Functions. In Image and Signal Processing for Remote Sensing VIII. Proceedings of SPIE Vol 4885, S.S. Serpico (Ed.), 215-225. Society of Photo-Optical Instrumentation Engineers).

Eklundh, L. and Olsson, L., 2003, Vegetation index trends for the African Sahel 1982-1999. Geophysical Research Letters, 30, 1430-1433.

Fensholt, R. and Proud, S.R. (2012), Evaluation of Earth Observation based global long term vegetation trends Ů Comparing GIMMS and MODIS global NDVI time series, Remote Sensing of Environment, 119, 131-147.

Gao, F., Morisette, J.T., Wolfe, R.E., Ederer, G., Pedelty, J., Masuoka, E., Myneni, R., Tan, B. and Nightingale, J., 2008, An algorithm to produce temporally and spatially continuous MODIS-LAI time-series, IEEE Geoscience and Remote Sensing Letters, 5.

Heumann, B.W., Seaquist, J. W., Eklundh, L. and Jönsson, P., 2007, AVHRR Derived Phenological Change in the Sahel and Soudan, Africa, 1982 - 2005. Remote Sensing of Environment, 108, 385-392.

Hickler, T., Eklundh, L., Seaquist, J., Smith, B., Ardö, J., Olsson, L., Sykes, M. and Sjöström, M., 2005, Precipitation controls Sahel greening trend. Geophysical Research Letters, 32, L21415.

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

Jamali, J., Jönsson, P., Eklundh, L., Ardö, J. and Seaquist, J., 2015, Detecting changes in vegetation trends using time series segmentation. Remote Sensing of Environment, 156, 182-195.

Jönsson, P. and Eklundh, L., 2002, Seasonality extraction by function fitting to time-series of satellite sensor data. IEEE Transactions on Geoscience and Remote Sensing, 40, 1824-1832.

Jönsson, P. and Eklundh, L., 2003, Seasonality extraction from time-series of satellite sensor data. In Frontiers of Remote Sensing Information Processing, C.H. Chen (Ed.), 487-500.(World Scientific Publishing).

Jönsson, P. and Eklundh, L., 2004, TIMESAT - a program for analysing time-series of satellite sensor data. Computers and Geosciences, 30, 833-845.

Jönsson, A.M., Eklundh, L., Hellström, M., Bärring, L. and Jönsson, P., 2010, Annual changes in MODIS vegetation indices of Swedish coniferous forests in relation to snow dynamics and tree phenology, Remote Sensing of Environment, 114, 2719-2730.

Kanzow, C., Yamashita, N., and Fukushima, M., 2002, Levenberg-Marquardt methods for constrained nonlinear equations with strong local convergence properties.

Le Page, Y., Oom, D., Silva, J.M.N., Jönsson, P. and Pereira, J.M.C., 2010, Seasonality of vegetation fires as modified by human action: observing the deviation from eco-climatic fire regimes. Global Ecology and Biogeography, 19, 575-588.

Madsen, K., Nielsen, H.B., and Tingleff, O., 2004, Methods for non-linear least squares problems, Informatics and Mathematical Modeling (IMM), Technical University of Denmark.

Nielsen, H.B., 1999, Damping parameter in Marquardt's method, Technical Report IMM-REP-1999-05, Informatics and Mathematical Modeling (IMM), Technical University of Denmark.

Nielsen, H.B., 2000, Separable nonlinear least squares, Technical Report IMM-REP-2000-01, Informatics and Mathematical Modeling (IMM), Technical University of Denmark (2000).

Olofsson, P. and Eklundh, L., 2007, Estimation of absorbed PAR across Scandinavia from satellite measurements. Part II: modeling and evaluating the fractional absorption. Remote Sensing of Environment, 110, 240-251.

Olofsson, P., Eklundh, L., Lagergren, F., Jönsson, P. and Lindroth, A., 2007, Estimating Net Primary Production for Scandinavian forests using data from Terra/MODIS. Advances in Space Research, 39, 125-130.

Olofsson, P., Lagergren, F., Lindroth, A., Lindström, J., Klemedtsson, L., Kutsch, W. and Eklundh, L., 2008, Toward Operational Remote Sensing of Forest Carbon Balance across Northern Europe. Biogeosciences, 5, 817-832.

Olsson, L., Eklundh, L. and Ardö, J., 2005, A recent greening of the Sahel - trends, patterns and potential causes. Journal of Arid Environments, 63, 556-566.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., 1994, Numerical Recipes in Fortran, Cambridge University Press.

Seaquist, J.W., Hickler, T., Eklundh, L., Ardö, J. and Heumann, B., 2009, Disentangling the effects of climate and people on Sahel vegetation dynamics. Biogeosciences, 6, 469-477.

Seaquist, J.W., Olsson, L., Ardö, J. and Eklundh, L., 2006, Broad-scale increase in NPP Quantified for the African Sahel, 1982-1999. International Journal of Remote Sensing, 27, 5115-5122.

Sjöström, M., Ardö, J., Eklundh, L., El-Tahir, B.A., El-Khidir, H.A.M., Hellström, M., Pilesjö, P. and Seaquist, J., 2009, Evaluation of satellite based indices for gross primary production estimates in a sparse savanna in the Sudan. Biogeosciences, 6, 129-138.

Schubert, P., Eklundh, L., Lund, M. and Nilsson, M., 2010, Estimating northern peatland CO2 exchange from MODIS time series data, Remote Sensing of Environment, 114, 1178-1189.

Schubert, P., Lagergren, F., Aurela, M., Christensen, T., Grelle, A., Heliasz, M., Klemedtsson, L., Lindroth, A., Pilegaard, K., Vesala, T., Eklundh, L., 2012, Modeling GPP in the Nordic forest landscape with MODIS time series data - comparison with the MODIS GPP product, Remote Sensing of Environment, vol. 126, 136-147.

Sjöström, M., Ardö, J., Arneth, A., Cappelaere, B., Eklundh, L., de Grandcourt, A., Kutsch, W. L., Merbold, L., Nouvellon, Y., Scholes, B., Seaquist, J. and Veenendaal, E. M., 2011, Exploring the potential of MODIS EVI for modeling gross primary production across African ecosystems. Remote Sensing of Environment, 115, 1081-1089.

Stisen, S., Sandholt, I., Norgaard, A., Fensholt, R. and Eklundh, L., 2007, Estimation of diurnal air temperature using MSG SEVIRI data in West Africa. Remote Sensing of Environment, 110, 262-274.

Tottrup, C., Schultz Rasmussen, M., Eklundh, L. and Jönsson, P., 2007, Mapping fractional forest cover across the highlands of mainland Southeast Asia using MODIS data and regression tree modelling. International Journal of Remote Sensing, 28, 23-46.

Verbesselt, J., Jönsson, P., Lhermitte, S., van Aardt, J. and Coppin, P., 2006, Evaluating satellite and climate data derived indices as fire risk indicators in savanna ecosystems. IEEE transactions of Geoscience and Remote Sensing, 44, 1622.

Verbesselt, J., Hyndman, R., Newnham, G. and Culvenor, D. (2010) Detecting trend and seasonal changes in satellite image time series. Remote Sensing of Environment, 114, 106-115

Yuan, H., Dai, Y., Xiao, Z., Ji, D. and Shangguan, W. (2011), Reprocessing the MODIS Leaf Area Index products for land surface and climate modelling, Remote Sensing of Environment, 115, 1171-1187.