Package `bfast`

February 19, 2015

Version 1.5.7
Date 2014-08-27
Title Breaks For Additive Season and Trend (BFAST)
Author Jan Verbesselt [aut, cre], Achim Zeileis [aut], Rob Hyndman [ctb]
Maintainer Jan Verbesselt <Jan.Verbesselt@wur.nl>
Description BFAST integrates the decomposition of time series into trend, seasonal, and remainder components with methods for detecting and characterizing abrupt changes within the trend and seasonal components. BFAST can be used to analyze different types of satellite image time series and can be applied to other disciplines dealing with seasonal or non-seasonal time series, such as hydrology, climatology, and econometrics. The algorithm can be extended to label detected changes with information on the parameters of the fitted piecewise linear models. BFAST monitoring functionality is added based on a paper that has been submitted to Remote Sensing of Environment.
BFAST monitor provides functionality to detect disturbance in near real-time based on BFAST-type models.
BFAST approach is flexible approach that handles missing data without interpolation.
Furthermore now different models can be used to fit the time series data and detect structural changes (breaks).

Depends R (>= 2.15.0)
Imports graphics, stats, strucchange, zoo, forecast, sp, raster

Suggests
License GPL (>= 2)

URL http://bfast.R-Forge.R-project.org/
LazyLoad yes
LazyData yes
Repository CRAN
Repository/R-Forge/Project bfast
Repository/R-Forge/Revision 464
**R topics documented:**

- bfast-package ................................. 2
- bfast ........................................ 3
- bfast01 ...................................... 6
- bfast01classify ............................. 9
- bfastmonitor ................................. 11
- bfastpp ...................................... 15
- bfastst ...................................... 17
- create16dayts ............................... 18
- dates ........................................ 19
- harvest ..................................... 19
- modisraster ................................. 20
- ndvi .......................................... 20
- plot.bfast .................................. 21
- simts ........................................ 22
- som .......................................... 22

**Index**

- bfast-package ................................. 2

---

**Description**

BFAST integrates the decomposition of time series into trend, seasonal, and remainder components with methods for detecting and characterizing abrupt changes within the trend and seasonal components. BFAST can be used to analyze different types of satellite image time series and can be applied to other disciplines dealing with seasonal or non-seasonal time series, such as hydrology, climatology, and econometrics. The algorithm can be extended to label detected changes with information on the parameters of the fitted piecewise linear models.

Additionally monitoring disturbances in BFAST-type models at the end of time series (i.e., in near real-time) is available: Based on a model for stable historical behaviour abnormal changes within newly acquired data can be detected. Different models are available for modeling the stable historical behavior. A season-trend model (with harmonic seasonal pattern) is used as a default in the regression modelling.
Details

The package contains:

- **bfast**: Main function for iterative decomposition and break detection as described in Verbesselt et al (2010ab).
- **bfastpp**: Data pre-processing for BFAST-type modeling.
- Functions for plotting and printing, see **bfast**.
- **simts**: Artificial example data set.
- **harvest**: NDVI time series of a P. radiata plantation that is harvested.
- **som**: NDVI time series of locations in the south of Somalia to illustrate the near real-time disturbance approach

Author(s)

Jan Verbesselt [aut, cre], Achim Zeileis [aut], Rob Hyndman [ctb], Rogier De Jong [ctb]

References


---

**bfast**

*Break Detection in the Seasonal and Trend Component of a Univariate Time Series*

Description

Iterative break detection in seasonal and trend component of a time series. Seasonal breaks is a function that combines the iterative decomposition of time series into trend, seasonal and remainder components with significant break detection in the decomposed components of the time series.

Usage

```r
bfast(Yt, h = 0.15, season = c("dummy", "harmonic", "none"),
     max.iter = NULL, breaks = NULL, hpc = "none", level = 0.05, type= "OLS-MOSUM")
```
Arguments

Yt   univariate time series to be analyzed. This should be an object of class "ts" with a frequency greater than one without NA's.

h   minimal segment size between potentially detected breaks in the trend model given as fraction relative to the sample size (i.e. the minimal number of observations in each segment divided by the total length of the timeseries.

season the seasonal model used to fit the seasonal component and detect seasonal breaks (i.e. significant phenological change). There are three options: "dummy", "harmonic", or "none" where "dummy" is the model proposed in the first Remote Sensing of Environment paper and "harmonic" is the model used in the second Remote Sensing of Environment paper (See paper for more details) and where "none" indicates that no seasonal model will be fitted (i.e. St = 0 ). If there is no seasonal cycle (e.g. frequency of the time series is 1) "none" can be selected to avoid fitting a seasonal model.

max.iter maximum amount of iterations allowed for estimation of breakpoints in seasonal and trend component.

breaks   integer specifying the maximal number of breaks to be calculated. By default the maximal number allowed by h is used.

hpc A character specifying the high performance computing support. Default is "none", can be set to "foreach". Install the "foreach" package for hpc support.

level numeric; threshold value for the sctest.efp test; if a length 2 vector is passed, the first value is used for the trend, the second for the seasonality

type character, indicating the type argument to efp

Details

To be completed.

Value

An object of the class "bfast" is a list with the following elements:

Yt equals the Yt used as input.

output is a list with the following elements (for each iteration):

Tt the fitted trend component
St the fitted seasonal component
Nt the noise or remainder component
Vt equals the deseasonalized data Yt - St for each iteration
bp.Vt output of the breakpoints function for the trend model
ci.Vt output of the breakpoints confint function for the trend model
Wt equals the detrended data Yt - Tt for each iteration
bp.Wt output of the breakpoints function for the seasonal model
ci.Wt output of the breakpoints confint function for the seasonal model

nobp is a list with the following elements:
nobp.Vt logical, TRUE if there are breakpoints detected
nobp.Wt logical, TRUE if there are breakpoints detected

magnitude magnitude of the biggest change detected in the trend component

Author(s)
Jan Verbesselt

References


See Also
plot.bfast for plotting of bfast() results.
breakpoints for more examples and background information about estimation of breakpoints in time series.

Examples
```r
## Not run:
rm(list = ls())
install.packages("bfast", repos="http://R-Forge.R-project.org", type = "source")
update.packages(checkBuilt=TRUE)
# make sure all your package are up to date
# and built correctly for your current R version

## End(Not run)

## Simulated Data
plot(simts) # stl object containing simulated NDVI time series
datats <- ts(rowSums(simts$time.series))
# sum of all the components (season,abrupt,remainder)
tsp(datats) <- tsp(simts$time.series) # assign correct time series attributes
plot(datats)

## Not run:
if (requireNamespace("forecast", quietly = TRUE)) {
  fit <- bfast(datats,h=0.15, season="dummy", max.iter=1)
  plot(fit,sim=simts)
  fit
  # prints out whether breakpoints are detected
  # in the seasonal and trend component
```


```r
} else {
    ## do something else not involving forecast related functions
    ## like seasonaldummy() and tsdisplay()
}

## End(Not run)

## Real data
## The data should be a regular ts() object without NA's
## See Fig. 8 b in reference
plot(harvest, ylab="NDVI")
# MODIS 16-day cleaned and interpolated NDVI time series

(rdist <- 10/length(harvest))
# ratio of distance between breaks (time steps) and length of the time series
## Not run:
if (requireNamespace("forecast", quietly = TRUE)) {
  fit <- bfast(harvest, h=rdist, season="harmonic", max.iter=1,breaks=2)
  plot(fit)
  ## plot anova and slope of the trend identified trend segments
  plot(fit, ANOVA=TRUE)
  ## plot the trend component and identify the break with
  ## the largest magnitude of change
  plot(fit, type="trend", largest=TRUE)

  ## plot all the different available plots
  plot(fit, type="all")

  ## output
  niter <- length(fit$output) # nr of iterations
  out <- fit$output[[niter]]
  # output of results of the final fitted seasonal and trend models and
  ## # nr of breakpoints in both.

  ## running bfast on yearly data
  t <- ts(as.numeric(harvest), frequency = 1, start = 2006)
  fit <- bfast(t, h = 0.23, season = "none", max.iter = 1)
  plot(fit)
  fit
}

## End(Not run)

```

---

**Checking for one major break in the time series**
**Description**

A function to select a suitable model for the data by choosing either a model with 0 or with 1 breakpoint.

**Usage**

```r
bfast01(data, formula = NULL,
        test = "OLS-MOSUM", level = 0.05, aggregate = all,
        trim = NULL, bandwidth = 0.15, functional = "max",
        order = 3, lag = NULL, slag = NULL, na.action = na.omit, stl = "none")
```

**Arguments**

- `data` A time series of class `ts`, or another object that can be coerced to such. The time series is processed by `bfastpp`. A time series of class `ts` can be prepared by a convenience function `bfastts` in case of daily, 10 or 16-daily time series.
- `formula` formula for the regression model. The default is intelligently guessed based on the arguments order/lag/slag i.e. `response ~ trend + harmon`, i.e., a linear trend and a harmonic season component. Other specifications are possible using all terms set up by `bfastpp`, i.e., season (seasonal pattern with dummy variables), lag (autoregressive terms), slag (seasonal autoregressive terms), or `xreg` (further covariates). See `bfastpp` for details.
- `test` character specifying the type of test(s) performed. Can be one or more of BIC, supLM, supF, OLS-MOSUM, ..., or any other test supported by `sctest.formula`
- `level` numeric. Significance for the `sctest.formula` performed.
- `aggregate` function that aggregates a logical vector to a single value. This is used for aggregating the individual test decisions from `test` to a single one.
- `trim` numeric. The minimal segment size passed to the `from` argument of the `Fstats` function.
- `bandwidth` numeric scalar from interval (0,1), functional. The bandwidth argument is passed to the `h` argument of the `sctest.formula`.
- `functional` arguments passed on to `sctest.formula`.
- `order` numeric. Order of the harmonic term, defaulting to 3.
- `lag` numeric. Order of the autoregressive term, by default omitted.
- `slag` numeric. Order of the seasonal autoregressive term, by default omitted.
- `na.action` arguments passed on to `bfastpp`.
- `stl` argument passed on to `bfastpp`.

**Details**

`bfast01` tries to select a suitable model for the data by choosing either a model with 0 or with 1 breakpoint. It proceeds in the following steps:

1. The data is preprocessed with `bfastpp` using the arguments `order/lag/slag/na.action/stl`.
2. A linear model with the given formula is fitted. By default a suitable formula is guessed based on the preprocessing parameters.
3. The model with 1 breakpoint is estimated as well where the breakpoint is chosen to minimize the 
segmented residual sum of squares.

4. A sequence of tests the null hypothesis of zero breaks is performed. Each test results in a decision 
for FALSE (no breaks) or TRUE (structural break(s)). The test decisions are then aggregated to a 
single decision (by default using all() but any() or some other function could also be used).

Available methods for the object returned include standard methods for linear models (coef, fitted, residuals, predict, AIC, BIC, logLik, deviance, nobs, model.matrix, model.frame), standard 
methods for breakpoints (breakpoints, breakdates), coercion to a zoo series with the decomposed 
components (as.zoo), and a plot method which plots such a zoo series along with the confidence 
interval (if the 1-break model is visualized). All methods take a 'breaks' argument which can either 
be 0 or 1. By default the value chosen based on the 'test' decisions is used.

Note that the different tests supported have power for different types of alternatives. Some tests 
(such as supLM/supF or BIC) assess changes in all coefficients of the model while residual-based 
tests (e.g., OLS-CUSUM or OLS-MOSUM) assess changes in the conditional mean. See Zeileis 
(2005) for a unifying view.

Value

`bfast01` returns a list of class "bfast01" with the following elements:

- **call**: the original function call.
- **data**: the data preprocessed by "bfastpp".
- **formula**: the model formulae.
- **breaks**: the number of breaks chosen based on the test decision (either 0 or 1).
- **test**: the individual test decisions.
- **breakpoints**: the optimal breakpoint for the model with 1 break.
- **model**: A list of two 'lm' objects with no and one breaks, respectively.

Author(s)

Achim Zeileis, Jan Verbesselt

References

Remote Sensing, 5, 1117–1133. [http://dx.doi.org/10.3390/rs5031117](http://dx.doi.org/10.3390/rs5031117)

Econometric Reviews, 24, 445–466. [http://dx.doi.org/10.1080/07474930500406053](http://dx.doi.org/10.1080/07474930500406053)

See Also

`bfastmonitor`, `breakpoints`
Examples

library(zoo)
## define a regular time series
ndvi <- as.ts(zoo(som$NDVI.a, som$Time))

## fit variations
bf1 <- bfast01(ndvi)
bf2 <- bfast01(ndvi, test = c("BIC", "OLS-MOSUM", "supLM"), aggregate = any)
bf3 <- bfast01(ndvi, test = c("OLS-MOSUM", "supLM"), aggregate = any, bandwidth = 0.11)

## inspect test decisions
bf1$test
bf1$breaks
bf2$test
bf2$breaks
bf3$test
bf3$breaks

## look at coefficients
coef(bf1)
coef(bf1, breaks = 0)
coef(bf1, breaks = 1)

## zoo series with all components
plot(as.zoo(ndvi))
plot(as.zoo(bf1, breaks = 1))
plot(as.zoo(bf2))
plot(as.zoo(bf3))

## leveraged by plot method
plot(bf1, regular = TRUE)
plot(bf2)
plot(bf2, plot.type = "multiple",
    which = c("response", "trend", "season"), screens = c(1, 1, 2))
plot(bf3)

bfast01classify

Change type analysis of the bfast01 function

Description

A function to determine the change type

Usage

bfast01classify(object, alpha = 0.05, pct_stable = NULL)
### Arguments

- **object**: `bfast01` object, i.e. the output of the `bfast01` function.
- **alpha**: threshold for significance tests, default 0.05
- **pct_stable**: threshold for segment stability, unit: percent change per unit time (0-100), default NULL

### Details

`bfast01classify`

### Value

`bfast01classify` returns a data.frame with the following elements:

- **flag_type**: Type of shift: (1) monotonic increase, (2) monotonic decrease, (3) monotonic increase (with positive break), (4) monotonic decrease (with negative break), (5) interruption: increase with negative break, (6) interruption: decrease with positive break, (7) reversal: increase to decrease, (8) reversal: decrease to increase
- **flag_significance**: SIGNIFICANCE FLAG: (0) both segments significant (or no break and significant), (1) only first segment significant, (2) only 2nd segment significant, (3) both segments insignificant (or no break and not significant)
- **flag_pct_stable**: STABILITY FLAG: (0) change in both segments is substantial (or no break and substantial), (1) only first segment substantial, (2) only 2nd segment substantial (3) both segments are stable (or no break and stable)

and also significance and percentage of both segments before and after the potentially detected break: "p_segment1", "p_segment2", "pct_segment1", "pct_segment2".

### Author(s)

Rogier de Jong, Jan Verbesselt

### References


### See Also

`bfast01`

### Examples

```r
library(zoo)
## define a regular time series
ndvi <- as.ts(zoo(som$NDVI.a, som$Time))
## fit variations
```
bfastmonitor

Near Real-Time Disturbance Detection Based on BFAST-Type Models

Description

Monitoring disturbances in time series models (with trend/season/regressor terms) at the end of time series (i.e., in near real-time). Based on a model for stable historical behaviour abnormal changes within newly acquired data can be detected. Different models are available for modeling the stable historical behavior. A season-trend model (with harmonic seasonal pattern) is used as a default in the regression modelling.

Usage

bfastmonitor(data, start, formula = response ~ trend + harmon, order = 3, lag = NULL, slag = NULL, history = c("ROC", "BP", "all"), type = "OLS-MOSUM", h = 0.25, end = 10, level = 0.05, hpc = "none", verbose = FALSE, plot = FALSE)

Arguments

data A time series of class ts, or another object that can be coerced to such. For seasonal components, a frequency greater than 1 is required.

start numeric. The starting date of the monitoring period. Can either be given as a float (e.g., 2000.5) or a vector giving period/cycle (e.g., c(2000, 7)).

formula formula for the regression model. The default is response ~ trend + harmon, i.e., a linear trend and a harmonic season component. Other specifications are possible using all terms set up by bfastpp, i.e., season (seasonal pattern with dummy variables), lag (autoregressive terms), slag (seasonal autoregressive terms), or xreg (further covariates). See bfastpp for details.

order numeric. Order of the harmonic term, defaulting to 3.

lag numeric. Order of the autoregressive term, by default omitted.

slag numeric. Order of the seasonal autoregressive term, by default omitted.

history specification of the start of the stable history period. Can either be a character, numeric, or a function. If character, then selection is possible between reverse-ordered CUSUM ("ROC", default), Bai and Perron breakpoint estimation ("BP"), or all available observations ("all"). If numeric, the start date can be specified in the same form as start. If a function is supplied it is called as history(formula, data) to compute a numeric start date.

type character specifying the type of monitoring process. By default, a MOSUM process based on OLS residuals is employed. See mefp for alternatives.
bfastmonitor

h numeric scalar from interval (0,1) specifying the bandwidth relative to the sample size in MOSUM/ME monitoring processes.

delta numeric. Maximum time (relative to the history period) that will be monitored (in MOSUM/ME processes). Default is 10 times the history period.

level numeric. Significance level of the monitoring (and ROC, if selected) procedure, i.e., probability of type I error.

hpc character specifying the high performance computing support. Default is "none", can be set to "foreach". See breakpoints for more details.

verbose logical. Should information about the monitoring be printed during computation?

plot logical. Should the result be plotted?

Details

bfastmonitor provides monitoring of disturbances (or structural changes) in near real-time based on a wide class of time series regression models with optional season/trend/autoregressive/covariate terms. See Verbesselt at al. (2011) for details.

Based on a given time series (typically, but not necessarily, with frequency greater than 1), the data is first preprocessed for regression modeling. Trend/season/autoregressive/covariate terms are (optionally) computed using bfastpp. Second, the data is split into a history and monitoring period (starting with start). Third, a subset of the history period is determined which is considered to be stable (see also below). Fourth, a regression model is fitted to the preprocessed data in the stable history period. Fifth, a monitoring procedure is used to determine whether the observations in the monitoring period conform with this stable regression model or whether a change is detected.

The regression model can be specified by the user. The default is to use a linear trend and a harmonic season: \texttt{response \sim trend + harmon}. However, all other terms set up by bfastpp can also be omitted/added, e.g., \texttt{response \sim 1} (just a constant), \texttt{response \sim season} (seasonal dummies for each period), etc. Further terms precomputed by bfastpp can be \texttt{lag} (autoregressive terms of specified order), \texttt{slag} (seasonal autoregressive terms of specified order), \texttt{xreg} (covariates, if data has more than one column).

For determining the size of the stable history period, various approaches are available. First, the user can set a start date based on subject-matter knowledge. Second, data-driven methods can be employed. By default, this is a reverse-ordered CUSUM test (ROC). Alternatively, breakpoints can be estimated (Bai and Perron method) and only the data after the last breakpoint are employed for the stable history. Finally, the user can also supply a function for his/her own data-driven method.

Value

bfastmonitor returns an object of class "bfastmonitor", i.e., a list with components as follows.

data original "ts" time series,
tsp preprocessed "data.frame" for regression modeling,
model fitted "lm" model for the stable history period,
mefp fitted "mefp" process for the monitoring period,
history start and end time of history period,
**monitor**

- **start and end time of monitoring period,**
- **breakpoint**
- **breakpoint detected (if any).**
- **magnitude**
- **median of the difference between the data and the model prediction in the monitoring period.**

**Author(s)**

Achim Zeileis, Jan Verbesselt

**References**


**See Also**

`monitor, mefp, breakpoints`

**Examples**

```r
## See Fig. 6 a and b in Verbesselt et al. (2011)
## for more information about the data time series and acknowledgements

library(zoo)
NDVIa <- as.ts(zoo(som$NDVI.a, som$Time))
plot(NDVIa)
## apply the bfast monitor function on the data
## start of the monitoring period is c(2010, 13)
## and the ROC method is used as a method to automatically identify a stable history
mona <- bfastmonitor(NDVIa, start = c(2010, 13))
mona
plot(mona)
## fitted season-trend model in history period
summary(mona$model)
## OLS-based MOSUM monitoring process
plot(mona$mefp, functional = NULL)
## the pattern in the running mean of residuals
## this illustrates the empirical fluctuation process
## and the significance of the detected break.

NDVIb <- as.ts(zoo(som$NDVI.b, som$Time))
plot(NDVIb)
monb <- bfastmonitor(NDVIb, start = c(2010, 13))
monb
plot(monb)
summary(monb$model)
plot(monb$mefp, functional = NULL)

## set the stable history period manually and use a 4th order harmonic model
```
```r
bfastmonitor(NDVIb, start = c(2010, 13),
  history = c(2008, 7), order = 4, plot = TRUE)

## just use a 6th order harmonic model without trend
mon <- bfastmonitor(NDVIb, formula = response ~ harmon,
  start = c(2010, 13), order = 6, plot = TRUE)
summary(mon$model)

## For more info
?bfastmonitor

## TUTORIAL for processing raster bricks (satellite image time series of 16-day NDVI images)
f <- system.file("extdata/modisraster.grd", package="bfast")
library("raster")
modisbrick <- brick(f)
data <- as.vector(modisbrick[[1]])
ndvi <- bfastts(data, dates, type = c("16-day"))
plot(ndvi/10000)

## derive median NDVI of a NDVI raster brick
medianNDVI <- calc(modisbrick, fun=function(x) median(x, na.rm = TRUE))
plot(medianNDVI)

## helper function to be used with the calc() function
xbfastmonitor <- function(x, dates) {
  ndvi <- bfastts(x, dates, type = c("16-day"))
  ndvi <- window(ndvi, end=c(2011,14))/10000
## delete end of the time to obtain a dataset similar to RSE paper (Verbesselt et al.,2012)
bfm <- bfastmonitor(data = ndvi, start=c(2010,12), history = c("ROC"))
return(cbind(bfm$breakpoint, bfm$magnitude))
}

## apply on one pixel for testing
ndvi <- bfastts(as.numeric(modisbrick[[1]])/10000, dates, type = c("16-day"))
plot(ndvi)

bfm <- bfastmonitor(data = ndvi, start=c(2010,12), history = c("ROC"))
bfm$magnitude
plot(bfm)
xbfastmonitor(modisbrick[[1]], dates) ## helper function applied on one pixel

## Not run:
## apply the bfastmonitor function onto a raster brick
library(raster)
timeofbreak <- calc(modisbrick, fun=function(x){
  res <- t(apply(x, 1, xbfastmonitor, dates))
return(res)
})
plot(timeofbreak) ## time of break and magnitude of change
plot(timeofbreak, 2) ## magnitude of change
```
### Description

Time series preprocessing for subsequent regression modeling. Based on a (seasonal) time series, a data frame with the response, seasonal terms, a trend term, (seasonal) autoregressive terms, and covariates is computed. This can subsequently be employed in regression models.

### Usage

```r
bfastpp(data, order = 3,
    lag = NULL, slag = NULL, na.action = na.omit,
    stl = c("none", "trend", "seasonal", "both"))
```

### Arguments

- **data**: A time series of class `ts`, or another object that can be coerced to such. For seasonal components, a frequency greater than 1 is required.
- **order**: numeric. Order of the harmonic term, defaulting to 3.
- **lag**: numeric. Orders of the autoregressive term, by default omitted.
- **slag**: numeric. Orders of the seasonal autoregressive term, by default omitted.
- **na.action**: function for handling NAs in the data (after all other preprocessing).
- **stl**: character. Prior to all other preprocessing, STL (season-trend decomposition via LOESS smoothing) can be employed for trend-adjustment and/or season-adjustment. The "trend" or "seasonal" component or both from `stl` are removed from each column in `data`. By default ("none"), no STL adjustment is used.

### Details

To facilitate (linear) regression models of time series data, `bfastpp` facilitates preprocessing and setting up regressor terms. It returns a `data.frame` containing the first column of the data as the response while further columns (if any) are used as covariates `xreg`. Additionally, a linear trend, seasonal dummies, harmonic seasonal terms, and (seasonal) autoregressive terms are provided.

Optionally, each column of `data` can be seasonally adjusted and/or trend-adjusted via STL (season-trend decomposition via LOESS smoothing) prior to preprocessing. The idea would be to capture season and/or trend nonparametrically prior to regression modelling.
Value

`bfastpp` returns a "data.frame" with the following variables (some of which may be matrices).

- **time**: numeric vector of time stamps,
- **response**: response vector (first column of data),
- **trend**: linear time trend (running from 1 to number of observations),
- **season**: factor indicating season period,
- **harmon**: harmonic seasonal terms (of specified order),
- **lag**: autoregressive terms (or orders `lag`, if any),
- **slag**: seasonal autoregressive terms (or orders `slag`, if any),
- **xreg**: covariate regressor (all columns of data except the first, if any).

Author(s)

Achim Zeileis

References


See Also

`bfastmonitor`

Examples

```r
## set up time series
library(zoo)
ndvi <- as.ts(zoo(cbind(a = som$NDVI.a, b = som$NDVI.b), som$Time))
ndvi <- window(ndvi, start = c(2006, 1), end = c(2009, 23))

## parametric season-trend model
d1 <- bfastpp(ndvi, order = 2)
d1lm <- lm(response ~ trend + harmon, data = d1)
summary(d1lm)

## autoregressive model (after nonparametric season-trend adjustment)
d2 <- bfastpp(ndvi, stl = "both", lag = 1:2)
d2lm <- lm(response ~ lag, data = d2)
summary(d2lm)
```
Create a regular time series object by combining data and date information.

Usage

bfastts(data, dates, type = c("irregular", "16-day", "10-day"))

Arguments

data A data vector

dates Optional input of dates for each measurement in the 'data' variable. In case the data is a irregular time series, a vector with 'dates' for each measurement can be supplied using this 'dates' variable. The irregular data will be linked with the dates vector to create daily regular time series with a frequency = 365. Extra days in leap years might cause problems. Please be carefull using this option as it is experimental. Feedback is welcome.

type ("irregular") indicates that the data is collected at irregular dates and as such will be converted to a daily time series. ("16-day")indicates that data is collected at a regular time interval (every 16-days e.g. like the MODIS 16-day data products). ("10-day") indicates that data is collected at a 10-day time interval of the SPOT VEGETATION (S10) product. Warning: Only use this function for the SPOT VEGETATION S10 time series, as for other 10-day time series a different approach might be required.

Details

bfastts create a regular time series

Value

bfastts returns an object of class "ts", i.e., a list with components as follows.

zz a regular "ts" time series with a frequency equal to 365 or 23 i.e. 16-day time series.

Author(s)

Achim Zeileis, Jan Verbesselt

See Also

monitor, mefp, breakpoints
Examples

```r
library("raster")
f <- system.file("extdata/modisraster.grd", package="bfast")
modisbrick <- brick(f)
ndvi <- bfastts(as.vector(modisbrick[1]), dates, type = c("16-day")) ## data of pixel 1
plot(ndvi/10000)
```

create16dayts

A helper function to create time series

Description

Time series creation

Usage

```
create16dayts(data, dates)
```

Arguments

```r
  data  A vector
  dates A vector ....
```

Author(s)

Achim Zeileis, Jan Verbesselt

See Also

```
bfastmonitor
```

Examples

```r
## set up a 16-day time series
ndvi <- create16dayts(modisraster[1], dates)
plot(ndvi)
```
dates

A vector with date information (a Datum type) to be linked with each NDVI layer within the modis raster brick (modisraster data set)

Description

dates is an object of class "Date" and contains the "Date" information to create a 16-day time series object.

Source


Examples

## see ?bfastmonitor for examples

harvest

16-day NDVI time series for a Pinus radiata plantation.

Description

A univariate time series object of class "ts". Frequency is set to 23 – the approximate number of observations per year.

Usage

data(harvest)

Source


Examples

plot(harvest,ylab='NDVI')
# References
citation("bfast")
modisraster

A raster brick of 16-day satellite image NDVI time series for a small subset in south eastern Somalia.

Description

A raster brick containing 16-day NDVI satellite images (MOD13C1 product).

Source


Examples

## see `?bfastmonitor`

ndvi

A random NDVI time series

Description

A univariate time series object of class "ts". Frequency is set to 24.

Usage

data(ndvi)

Examples

plot(ndvi)
Methods for objects of class "bfast".

Description

Plot methods for objects of class "bfast".

Usage

```r
## S3 method for class 'bfast'
plot(x, type = c("components", "all", "data", "seasonal", "trend", "noise"), sim = NULL, largest=FALSE, main, ANOVA = FALSE, ...)
```

Arguments

- `x`  
  `bfast` object
- `type`  
  Indicates the type of plot. See details.
- `sim`  
  Optional `stl` object containing the original components used when simulating `x`.
- `largest`  
  If TRUE, show the largest jump in the trend component.
- `ANOVA`  
  if TRUE Derive Slope and Significance values for each identified trend segment
- `main`  
  an overall title for the plot.
- `...`  
  further arguments passed to the `plot` function.

Details

This function creates various plots to demonstrate the results of a bfast decomposition. The type of plot shown depends on the value of `type`.

- `components`  
  Shows the final estimated components with breakpoints.
- `all`  
  Plots the estimated components and breakpoints from all iterations.
- `data`  
  Just plots the original time series data.
- `seasonal`  
  Shows the trend component including breakpoints.
- `trend`  
  Shows the trend component including breakpoints.
- `noise`  
  Plots the noise component along with its acf and pacf.

If `sim` is not NULL, the components used in simulation are also shown on each graph.

Author(s)

Jan Verbesselt, Rob Hyndman and Rogier De Jong

Examples

```r
## See \code{\link[bfast]{bfast}} for examples.
```
Simulated seasonal 16-day NDVI time series

**Description**

*simts* is an object of class "stl" and consists of seasonal, trend (equal to 0) and noise components. The simulated noise is typical for remotely sensed satellite data.

**Usage**

data(simts)

**Source**


**Examples**

plot(simts)
# References
citation("bfast")

Two 16-day NDVI time series from the south of Somalia

**Description**

*som* is a dataframe containing time and two NDVI time series to illustrate how the monitoring approach works.

**Usage**

data(som)

**Source**

Needs to be added.

**Examples**

## first define the data as a regular time series (i.e. ts object)
library(zoo)
NDVI <- as.ts(zoo(som$NDVI,b,som$time))
plot(NDVI)
Index

*Topic **datasets**
  dates, 19
  harvest, 19
  modisraster, 20
  ndvi, 20
  simts, 22
  som, 22

*Topic **ts, bfast01**
  bfast01classify, 9

*Topic **ts**
  bfast, 3
  bfast-package, 2
  bfast01, 6
  bfastmonitor, 11
  bfastpp, 15
  bfaststs, 17
  create16dayts, 18
  harvest, 19
  modisraster, 20
  ndvi, 20
  plot, 21
  plot.bfast, 5, 21

  sctest.efp, 4
  sctest.formula, 7
  simts, 3, 22
  som, 3, 22
  stl, 15, 21

  ts, 7, 11, 15

bfast, 3, 3, 21
bfast-package, 2
bfast01, 6, 10
bfast01classify, 9
bfastmonitor, 3, 8, 11, 16, 18
bfastpp, 3, 7, 11, 12, 15
bfaststs, 7, 17
breakpoints, 4, 5, 8, 12, 13, 17

create16dayts, 18
dates, 19
efp, 4
Fstats, 7
harvest, 3, 19
mefp, 11, 13, 17
modisraster, 20
monitor, 13, 17
ndvi, 20
plot, 21
plot.bfast, 5, 21

sctest.efp, 4
sctest.formula, 7
simts, 3, 22
som, 3, 22
stl, 15, 21

ts, 7, 11, 15