Package 'bfast'

April 1, 2026

Version 1.7.2

Title Breaks for Additive Season and Trend

Description 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 described
in Verbesselt et al. (2010) <doi:10.1016/j.rse.2009.08.014>. 'BFAST monitor'
provides functionality to detect disturbance in near real-time based on 'BFAST'-
type models, and is described in Verbesselt et al. (2012) <doi:10.1016/j.rse.2012.02.022>.
'BFAST Lite' approach is a flexible approach that handles missing data
without interpolation, and will be described in an upcoming paper.
Furthermore, different models can now be used to fit the
time series data and detect structural changes (breaks).

Depends R (>= 3.0.0), strucchangeRcpp (>= 1.5-4)

Imports graphics, stats, zoo, forecast, Rcpp (>= 0.12.7), Rdpack (>=
0.7)

Suggests MASS, sfsmisc, stlplus, terra

License GPL (>= 2)

URL https://bfast2.github.io/

BugReports https://github.com/bfast2/bfast/issues

LazyLoad yes

LazyData yes

LinkingTo Rcpp

RoxygenNote 7.3.3

RdMacros Rdpack

Encoding UTF-8

NeedsCompilation yes

Author Jan Verbesselt [aut],
Dainius Masiliūnas [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5654-1277>),
Achim Zeileis [aut],
Rob Hyndman [ctb],
Marius Appel [aut],
Martin Jung [ctb],
Andrei Mîrț [ctb] (ORCID: <https://orcid.org/0000-0003-3654-2090>),
Paulo Negri Bernardino [ctb],
Dongdong Kong [ctb] (ORCID: <https://orcid.org/0000-0003-1836-8172>)

Maintainer Dainius Masiliūnas <pastas4@gmail.com>

Repository CRAN

Date/Publication 2026-04-01 05:10:46 UTC

Contents

bfast-package 2
.bfast_cpp_closestfrom 4
bfast 5
bfast01 8
bfast01classify 11
bfastlite 13
bfastmonitor 16
bfastpp 20
bfastts 22
create16dayts-deprecated 24
dates 24
harvest 25
modisraster 25
ndvi 26
plot.bfast 26
plot.bfastlite 27
setoptions 28
simts 29
som 29

Index **30**

`bfast-package` *Breaks For Additive Season and Trend (BFAST)*

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 regresssion modelling.

Details

The package contains:

- `bfast()`: Main function for iterative decomposition and break detection as described in Verbesselt et al (2010a,b).
- `bfastlite()`: lightweight and fast detection of all breaks in a time series using a single iteration with all components at once.
- `bfastmonitor()`: Monitoring approach for detecting disturbances in near real-time (see Verbesselt et al. 2012).
- `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

Package options

bfast uses the following options to modify the default behaviour:

- `bfast.prefer_matrix_methods`: logical value defining whether methods should try to use the design matrix instead of the formula and a dataframe whenever possible. This can avoid expensive repeated calls of `model.matrix` and `model.frame` and make model fitting faster using `lm.fit`.
- `bfast.use_bfastts_modifications`: logical value defining whether a faster version of `bfastts()` should be used.
- `strucchange.use_armadillo`: logical value defining whether to use C++ optimised code paths in strucchangeRcpp.

By default, all three are enabled. See `set_fallback_options()` for a convenient interface for setting them all off for debugging purposes.

References

Verbesselt J, Zeileis A, Herold M (2012). "Near real-time disturbance detection using satellite image time series." *Remote Sensing of Environment*, **123**, 98–108. ISSN 0034-4257, doi:10.1016/j.rse.2012.02.022.

Verbesselt J, Hyndman R, Newnham G, Culvenor D (2010). "Detecting trend and seasonal changes in satellite image time series." *Remote Sensing of Environment*, **114**(1), 106–115. ISSN 0034-4257, doi:10.1016/j.rse.2009.08.014.

Verbesselt J, Hyndman R, Zeileis A, Culvenor D (2010). "Phenological change detection while accounting for abrupt and gradual trends in satellite image time series." *Remote Sensing of Environment*, **114**(12), 2970–2980. ISSN 0034-4257, doi:10.1016/j.rse.2010.08.003.

`.bfast_cpp_closestfrom` *For all elements of a vector a, find the closest elements in a vector B and returns resulting indexes*

Description

For all elements of a vector a, find the closest elements in a vector B and returns resulting indexes

Usage

```
.bfast_cpp_closestfrom(a, b, twosided)
```

Arguments

`a`	numeric vector, ordered
`b`	numeric vector, ordered
`twosided`	logical value, if false, indexes will always point to elements in b that are less than or equal to elements in a but not greater than.

Value

integer vector of the same size as a with elements represnting indexes pointing to closest values in b

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

```
bfast(
  Yt,
  h = 0.15,
  season = c("dummy", "harmonic", "none"),
  max.iter = 10,
  breaks = NULL,
  hpc = "none",
  level = 0.05,
  decomp = c("stl", "stlplus"),
  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.
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	either an integer specifying the number of breaks to compute, or a string specifying a statistic with which to compute the optimal number of breakpoints (see strucchangeRcpp::breakpoints() for more information). If one value is given, it is used for both the trend and seasonal components, and if two are

given, the first one is considered to be for the trend and the second for the seasonal component.

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
decomp	"stlplus" or "stl": the function to use for decomposition. `stl` can handle sparse time series (1 < frequency < 4), `stlplus` can handle `NA` values in the time series.
type	character, indicating the type argument to efp
...	additional arguments passed to `stlplus::stlplus()`, if its usage has been enabled, otherwise ignored.

Details

The algorithm decomposes the input time series `Yt` into three components: trend, seasonality and remainder, using the function defined by the `decomp` parameter. Then each component is checked for at least one significant break using `strucchangeRcpp::efp()`, and if there is one, `strucchangeRcpp::breakpoints()` is run on the component. The result allows differentiating between breaks in trend and seasonality.

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. Note that the output breakpoints are index numbers of `na.o`
`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. Note that the output breakpoints are index numbers of `na`
`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 no breakpoints detected
`nobp.Wt`	logical, TRUE if there are no breakpoints detected

`Magnitude`	magnitude of the biggest change detected in the trend component
`Time`	timing of the biggest change detected in the trend component

Author(s)

Jan Verbesselt

References

Verbesselt J, Hyndman R, Newnham G, Culvenor D (2010). "Detecting trend and seasonal changes in satellite image time series." *Remote Sensing of Environment*, **114**(1), 106–115. ISSN 0034-4257, doi:10.1016/j.rse.2009.08.014.

Verbesselt J, Hyndman R, Zeileis A, Culvenor D (2010). "Phenological change detection while accounting for abrupt and gradual trends in satellite image time series." *Remote Sensing of Environment*, **114**(12), 2970–2980. ISSN 0034-4257, doi:10.1016/j.rse.2010.08.003.

See Also

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

Examples

```
## 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)

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

## 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

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

## handling missing values with stlplus
(NDVIa <- as.ts(zoo::zoo(som$NDVI.a, som$Time)))
fit <- bfast(NDVIa, season = "harmonic", max.iter = 1, decomp = "stlplus")
plot(fit)
fit
```

---

bfast01 *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

```
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,
  reg = c("lm", "rlm"),
  stl = "none",
  sbins = 1
)
```

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 autoregressiv 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 mimimal 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
reg	whether to use OLS regression lm() or robust regression MASS::rlm().
stl	argument passed on to bfastpp
sbins	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/sbins.
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 for 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

De Jong R, Verbesselt J, Zeileis A, Schaepman ME (2013). "Shifts in Global Vegetation Activity Trends." *Remote Sensing*, **5**(3), 1117–1133. ISSN 2072-4292, doi:10.3390/rs5031117.

Zeileis A (2005). "A Unified Approach to Structural Change Tests Based on ML Scores, F Statistics, and OLS Residuals." *Econometric Reviews*, **24**(4), 445–466. ISSN 0747-4938, doi: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,
  typology = c("standard", "drylands")
)
```

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
`typology`	classification legend to use: `standard` refers to the original legend as used in De Jong et al. (2013), `drylands` refers to the legend used in Bernardino et al. (2020).

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

Bernardino PN, De Keersmaecker W, Fensholt R, Verbesselt J, Somers B, Horion S (2020). "Global-scale characterization of turning points in arid and semi-arid ecosystem functioning." *Global Ecology and Biogeography*, **29**(7), 1230–1245. doi:10.1111/geb.13099.

De Jong R, Verbesselt J, Zeileis A, Schaepman ME (2013). "Shifts in Global Vegetation Activity Trends." *Remote Sensing*, **5**(3), 1117–1133. ISSN 2072-4292, doi:10.3390/rs5031117.

See Also

`bfast01`

Examples

```
library(zoo)
## define a regular time series
ndvi <- as.ts(zoo(som$NDVI.a, som$Time))
## fit variations
bf1 <- bfast01(ndvi)
bfast01classify(bf1, pct_stable = 0.25)
```

bfastlite *Detect multiple breaks in a time series*

Description

A combination of bfastpp and breakpoints to do light-weight detection of multiple breaks in a time series while also being able to deal with NA values by excluding them via bfastpp.

Usage

```
bfastlite(
  data,
  formula = response ~ trend + harmon,
  order = 3,
  breaks = "LWZ",
  lag = NULL,
  slag = NULL,
  na.action = na.omit,
  stl = c("none", "trend", "seasonal", "both"),
  decomp = c("stl", "stlplus"),
  sbins = 1,
  h = 0.15,
  level = 0,
  type = "OLS-MOSUM",
  ...
)

bfast0n(
  data,
  formula = response ~ trend + harmon,
  order = 3,
  breaks = "LWZ",
  lag = NULL,
  slag = NULL,
  na.action = na.omit,
  stl = c("none", "trend", "seasonal", "both"),
  decomp = c("stl", "stlplus"),
  sbins = 1,
  h = 0.15,
  level = 0,
  type = "OLS-MOSUM",
  ...
)
```

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.

formula	a symbolic description for the model in which breakpoints will be estimated.
order	numeric. Order of the harmonic term, defaulting to 3.
breaks	either a positive integer specifying the maximal number of breaks to be calculated, or a string specifying the information criterion to use to automatically determine the optimal number of breaks (see also logLik). "all" means the maximal number allowed by h is used. In breakpoints.breakpointsfull, NULL extracts the optimal number of breaks as determined by the initial breakpoints call.
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.
decomp	"stlplus" or "stl": use the NA-tolerant decomposition package or the reference package (which can make use of time series with 2-3 observations per year)
sbins	numeric. Controls the number of seasonal dummies. If integer > 1, sets the number of seasonal dummies to use per year. If <= 1, treated as a multiplier to the number of observations per year, i.e. ndummies = nobs/year * sbins.
h	minimal segment size either given as fraction relative to the sample size or as an integer giving the minimal number of observations in each segment.
level	numeric; threshold value for the sctest.efp test for at least one break in a time series, to save processing time for no-change areas. The test is skipped if set to <= 0.
type	character, indicating the type argument to efp.
...	Additional arguments to breakpoints.

Value

An object of class bfastlite, with three elements:

breakpoints	output from breakpoints, containing information about the estimated breakpoints.
data_pp	preprocessed data as output by bfastpp.
sctest	output from sctest.efp, contatining information about the likelihood of the time series having at least one break.

Author(s)

Dainius Masiliunas, Jan Verbesselt

References

Masiliūnas D, Tsendbazar N, Herold M, Verbesselt J (2021). "BFAST Lite: A Lightweight Break Detection Method for Time Series Analysis." *Remote Sensing*, **13**(16), 3308. doi:10.3390/rs13163308.

Examples

```
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)

# Detect breaks
bp = bfastlite(datats)

# Default method of estimating breakpoints
bp[["breakpoints"]][["breakpoints"]]

# Plot
plot(bp)

# Custom method of estimating number of breaks (request 2 breaks)
strucchangeRcpp::breakpoints(bp[["breakpoints"]], breaks = 2)

# Plot including magnitude based on RMSD for the cos1 component of harmonics
plot(bp, magstat = "RMSD", magcomp = "harmoncos1", breaks = 2)

# Try with a structural change test
bp <- bfastlite(datats, level=0.05)
print(bp)
plot(bp)

# Details of the structural change test with the type RE
bfastlite(datats, level=0.05, type="RE")$sctest


## Run bfastlite() on a raster
f <- system.file("extdata/modisraster.tif", package="bfast")
modisbrick <- terra::rast(f)

# Run on pixel 10
data <- unlist(modisbrick[10])
ndvi <- bfastts(data, dates, type = c("16-day"))
bfl <- bfastlite(ndvi, breaks = "BIC")
# Get max magnitude by RMSD
max(magnitude(bfl[["breakpoints"]])$Mag[,"RMSD"])

# Wrapper function
bflSpatial <- function(pixels)
{
    ts <- bfastts(pixels, dates, type = c("16-day"))
    bfl <- bfastlite(ts, breaks="BIC")
    bp <- bfl[["breakpoints"]]
    # Return number of breakpoints and max breakpoint magnitude
    if (length(bp[["breakpoints"]]) == 1 && is.na(bp[["breakpoints"]]))
        return(c(0, 0))
```

```
    return(c(length(bp[["breakpoints"]]), max(magnitude(bp)$Mag[,"RMSD"])))
}

# Run function on each raster pixel
rastbfl <- terra::app(modisbrick, bflSpatial)
terra::plot(rastbfl)
```

---

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 regresssion 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 = c(0.05, 0.05),
  hpc = "none",
  verbose = FALSE,
  plot = FALSE,
  sbins = 1
)
```

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.
h	numeric scalar from interval (0,1) specifying the bandwidth relative to the sample size in MOSUM/ME monitoring processes.
end	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 vector. Significance levels 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?
sbins	numeric. Number of seasonal dummies, passed to bfastpp.

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: response ~ trend + harmon. However, all other terms set up by bfastpp can also be omitted/added, e.g., response ~ 1 (just a constant), response ~ season (seasonal dummies for

each period), etc. Further terms precomputed by `bfastpp` can be `lag` (autoregressive terms of specified order), `slag` (seasonal autoregressive terms of specified order), `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,
`tspp`	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

Verbesselt J, Zeileis A, Herold M (2012). "Near real-time disturbance detection using satellite image time series." *Remote Sensing of Environment*, **123**, 98–108. ISSN 0034-4257, doi:10.1016/j.rse.2012.02.022.

See Also

`monitor`, `mefp`, `breakpoints`

Examples

```
NDVIa <- as.ts(zoo::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
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$model)
AIC(mon$model)

## use a custom number of seasonal dummies (11/yr) instead of harmonics
mon <- bfastmonitor(NDVIb, formula = response ~ season,
    start = c(2010, 13), sbins = 11, plot = TRUE)
summary(mon$model)
AIC(mon$model)


## Example for processing raster bricks (satellite image time series of 16-day NDVI images)
f <- system.file("extdata/modisraster.tif", package="bfast")
modisbrick <- terra::rast(f)
data <- unlist(modisbrick[1])
ndvi <- bfastts(data, dates, type = c("16-day"))
plot(ndvi/10000)

## derive median NDVI of a NDVI raster brick
medianNDVI <- terra::app(modisbrick, fun = "median")
terra::plot(medianNDVI)

## helper function to be used with the app() function
xbfastmonitor <- function(x, timestamps = dates) {
ndvi <- bfastts(x, timestamps, 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(c(breakpoint = bfm$breakpoint, magnitude = bfm$magnitude))
}

## apply on one pixel for testing
```

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

## apply the bfastmonitor function onto a raster brick
timeofbreak <- terra::app(modisbrick, fun=xbfastmonitor)

terra::plot(timeofbreak) ## time of break and magnitude of change
terra::plot(timeofbreak,2) ## magnitude of change
```

---

bfastpp *Time Series Preprocessing for BFAST-Type Models*

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

```
bfastpp(
  data,
  order = 3,
  lag = NULL,
  slag = NULL,
  na.action = na.omit,
  stl = c("none", "trend", "seasonal", "both"),
  decomp = c("stl", "stlplus"),
  sbins = 1
)
```

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.

decomp	"stlplus" or "stl": use the NA-tolerant decomposition package or the reference package (which can make use of time series with 2-3 observations per year)
sbins	numeric. Controls the number of seasonal dummies. If integer > 1, sets the number of seasonal dummies to use per year. If <= 1, treated as a multiplier to the number of observations per year, i.e. ndummies = nobs/year * sbins.

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

If no formula is provided, 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).

If a formula is given, bfastpp returns a list with components X, y, and t, where X is the design matrix of the model, y is the response vector, and t represents the time of observations. X will only contain variables that occur in the formula. Columns of X have names as decribed above.

Author(s)

Achim Zeileis

References

Verbesselt J, Zeileis A, Herold M (2012). "Near real-time disturbance detection using satellite image time series." *Remote Sensing of Environment*, **123**, 98–108. ISSN 0034-4257, doi:10.1016/j.rse.2012.02.022.

See Also

bfastmonitor

Examples

```
## set up time series
ndvi <- as.ts(zoo::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)
# plot visually (except season, as it's a factor)
plot(zoo::read.zoo(d1)[,-3],
  # Avoid clipping plots for pretty output
  ylim = list(c(min(d1[,2]), max(d1[,2])),
              c(min(d1[,3]), max(d1[,3])),
              c(-1, 1), c(-1, 1), c(-1, 1), c(-1, 1),
              c(min(d1[,6]), max(d1[,6]))
       ))

## autoregressive model (after nonparametric season-trend adjustment)
d2 <- bfastpp(ndvi, stl = "both", lag = 1:2)
d2lm <- lm(response ~ lag, data = d2)
summary(d2lm)

## use the lower level lm.fit function
d3 <- bfastpp(ndvi, stl = "both", lag = 1:2)
d3mm <- model.matrix(response ~ lag, d3)
d3lm <- lm.fit(d3mm, d3$response)
d3lm$coefficients
```

`bfastts` *Create a regular time series object by combining data and date information*

Description

Create a regular time series object by combining measurements (data) and time (dates) information.

Usage

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

Arguments

`data`	A data vector or matrix where columns represent variables
`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 careful 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

```
# 16-day time series (i.e. MODIS)
timedf <- data.frame(y = som$NDVI.b, dates = dates[1:nrow(som)])
bfastts(timedf$y, timedf$dates, type = "16-day")

# Irregular
head(bfastts(timedf$y, timedf$dates, type = "irregular"), 50)


# Example of use with a raster
f <- system.file("extdata/modisraster.tif", package="bfast")
modisbrick <- terra::rast(f)
ndvi <- bfastts(unlist(modisbrick[1]), dates, type = c("16-day")) ## data of pixel 1
plot(ndvi/10000)

# Time series of 4 pixels
modis_ts = t(modisbrick[1:4])
# Data with multiple columns, 2-4 are external regressors
ndvi <- bfastts(modis_ts, dates, type = c("16-day"))
plot(ndvi/10000)
```

create16dayts-deprecated

A helper function to create time series

Description

A deprecated alias to bfastts. Please use `bfastts(type="16-day")` instead.

Usage

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

Arguments

`data`	Passed to bfastts.
`dates`	Passed to bfastts.

Author(s)

Achim Zeileis, Jan Verbesselt

See Also

`bfastmonitor`

`bfast-deprecated`

dates

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

Verbesselt J, Zeileis A, Herold M (2012). "Near real-time disturbance detection using satellite image time series." *Remote Sensing of Environment*, **123**, 98–108. ISSN 0034-4257, doi:10.1016/j.rse.2012.02.022.

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.

Source

Verbesselt J, Hyndman R, Newnham G, Culvenor D (2010). "Detecting trend and seasonal changes in satellite image time series." *Remote Sensing of Environment*, **114**(1), 106–115. ISSN 0034-4257, doi:10.1016/j.rse.2009.08.014.

Examples

```
plot(harvest,ylab='NDVI')
```

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

Description

A Cloud-Optimised GeoTIFF containing 16-day NDVI satellite images (MOD13C1 product).

Source

Verbesselt J, Zeileis A, Herold M (2012). "Near real-time disturbance detection using satellite image time series." *Remote Sensing of Environment*, **123**, 98–108. ISSN 0034-4257, doi:10.1016/j.rse.2012.02.022.

Examples

```
## see ?bfastmonitor
```

ndvi *A random NDVI time series*

Description

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

Examples

```
plot(ndvi)
```

plot.bfast *Methods for objects of class "bfast".*

Description

Plot methods for objects of class "bfast".

Usage

```
## 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.
main	an overall title for the plot.
ANOVA	if TRUE Derive Slope and Significance values for each identified trend segment
...	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 seasonal 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.

Value

No return value, called for side effects.

Author(s)

Jan Verbesselt, Rob Hyndman and Rogier De Jong

Examples

```
## See \code{\link[bfast]{bfast}} for examples.
```

---

plot.bfastlite *Plot the time series and results of BFAST Lite*

Description

The black line represents the original input data, the green line is the fitted model, the blue lines are the detected breaks, and the whiskers denote the magnitude (if `magstat` is specified).

Usage

```
## S3 method for class 'bfastlite'
plot(x, breaks = NULL, magstat = NULL, magcomp = "trend", ...)
```

Arguments

`x`	bfastlite object from `bfastlite()`
`breaks`	number of breaks or optimal break selection method, see `strucchangeRcpp::breakpoints()`
`magstat`	name of the magnitude column to plot (e.g. `RMSD`, `MAD`, `diff`), see the `Mag` component of `strucchangeRcpp::magnitude.breakpointsfull()`
`magcomp`	name of the component (i.e. column in `x$data_pp`) to plot magnitudes of
`...`	other parameters to pass to `plot()`

Value

Nothing, called for side effects.

`setoptions` *Set package options with regard to computation times*

Description

These functions set options of the bfast and strucchangeRcpp packages to enable faster computations. By default (`set_default_options`), these optimizations are enabled. Notice that only some functions of the `bfast` package make use of these options. `set_fast_options` is an alias for `set_default_options`.

Usage

```
set_default_options()

set_fast_options()

set_fallback_options()
```

Value

A list of modified options and their new values.

Examples

```
# run bfastmonitor with different options and compare computation times
library(zoo)
NDVIa <- as.ts(zoo(som$NDVI.a, som$Time))

set_default_options()
## Not run:
system.time(replicate(100,  bfastmonitor(NDVIa, start = c(2010, 13))))

## End(Not run)

set_fallback_options()
## Not run:
system.time(replicate(100,  bfastmonitor(NDVIa, start = c(2010, 13))))

## End(Not run)
```

simts *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.

Source

Verbesselt J, Hyndman R, Newnham G, Culvenor D (2010). "Detecting trend and seasonal changes in satellite image time series." *Remote Sensing of Environment*, **114**(1), 106–115. ISSN 0034-4257, doi:10.1016/j.rse.2009.08.014.

Examples

```
plot(simts)
```

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

Description

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

Source

Verbesselt J, Zeileis A, Herold M (2012). "Near real-time disturbance detection using satellite image time series." *Remote Sensing of Environment*, **123**, 98–108. ISSN 0034-4257, doi:10.1016/j.rse.2012.02.022.

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

∗ **bfast01**
bfast01classify, 11
∗ **datasets**
dates, 24
harvest, 25
modisraster, 25
ndvi, 26
simts, 29
som, 29
∗ **ts**
bfast, 5
bfast-package, 2
bfast01, 8
bfast01classify, 11
bfastmonitor, 16
bfastpp, 20
bfastts, 22
create16dayts-deprecated, 24
harvest, 25
modisraster, 25
ndvi, 26
plot.bfast, 26
.bfast_cpp_closestfrom, 4

bfast, 5, *26*
bfast(), *3*
bfast-package, 2
bfast01, 8, *11, 12*
bfast01classify, 11
bfast0n *(*bfastlite*)*, 13
bfastlite, 13
bfastlite(), *3*, *27*
bfastmonitor, *10*, 16, *21*, *24*
bfastmonitor(), *3*
bfastpp, *8, 9*, *13, 14*, *17*, 20
bfastpp(), *3*
bfastts, *8*, 22
bfastts(), *3*
breakpoints, *6, 7*, *10*, *13, 14*, *17, 18*, *23*

create16dayts-deprecated, 24

dates, 24

efp, *6*, *14*

Fstats, *9*

harvest, *3*, 25

lm(), *9*
logLik, *14*

MASS::rlm(), *9*
mefp, *17, 18*, *23*
modisraster, 25
monitor, *18*, *23*

ndvi, 26

plot, *26*
plot(), *27*
plot.bfast, *7*, 26
plot.bfastlite, 27

sctest.efp, *6*, *14*
sctest.formula, *9*
set_default_options *(*setoptions*)*, 28
set_fallback_options *(*setoptions*)*, 28
set_fallback_options(), *3*
set_fast_options *(*setoptions*)*, 28
setoptions, 28
simts, *3*, 29
som, *3*, 29
stl, *14*, *20*, *26*
stlplus::stlplus(), *6*
strucchangeRcpp::breakpoints(), *5, 6*, *27*
strucchangeRcpp::efp(), *6*
strucchangeRcpp::magnitude.breakpointsfull(), *27*

ts, *8*, *13*, *16*, *20*