## loadZbfast {bfast} R Documentation

## 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 = NULL, breaks = NULL, hpc ="none")

### 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. |

### 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](http://127.0.0.1:18458/library/strucchange/html/breakpoints.html) function for the trend model | | ci.Vt | output of the [breakpoints](http://127.0.0.1:18458/library/strucchange/html/breakpoints.html) confint function for the trend model | | Wt | equals the detrended data Yt - Tt for each iteration | | bp.Vt | output of the [breakpoints](http://127.0.0.1:18458/library/strucchange/html/breakpoints.html) function for the seasonal model | | ci.Vt | output of the [breakpoints](http://127.0.0.1:18458/library/strucchange/html/breakpoints.html) 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 |
| Time | timing of the biggest change detected in the trend component |

### Author(s) : Jan Verbesselt

### References :

* Verbesselt, J., R. Hyndman, G. Newnham, and D. Culvenor (2009). Detecting trend and seasonal changes in satellite image time series. *Remote Sensing of Environment*. <http://dx.doi.org/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, 2970-2980. <http://dx.doi.org/10.1016/j.rse.2010.08.003>. Or see <http://bfast.r-forge.r-project.org/>.

### See Also

[plot.bfast](http://127.0.0.1:18458/library/bfast/html/plot.bfast.html) for plotting of bfast() results.   
[breakpoints](http://127.0.0.1:18458/library/strucchange/html/breakpoints.html) 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(fit,type="trend",largest=TRUE)

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.

# References

citation("bfast")

# For more info

?bfast

**function (Yt, h = 0.15, season = c("dummy", "harmonic", "none"),**

**max.iter = NULL, breaks = NULL, hpc = "none")**

**{**

**season <- match.arg(season)**

**ti <- time(Yt)**

**f <- frequency(Yt)**

**if (class(harvest) != "ts")**

**stop("Not a time series object")**

**output <- list()**

**St <- stl(Yt, "periodic")$time.series[, "seasonal"]**

**Tt <- 0**

**if (season == "harmonic") {**

**w <- 1/23**

**tl <- 1:length(Yt)**

**co <- cos(2 \* pi \* tl \* w)**

**si <- sin(2 \* pi \* tl \* w)**

**co2 <- cos(2 \* pi \* tl \* w \* 2)**

**si2 <- sin(2 \* pi \* tl \* w \* 2)**

**co3 <- cos(2 \* pi \* tl \* w \* 3)**

**si3 <- sin(2 \* pi \* tl \* w \* 3)**

**smod <- Wt ~ co + si + co2 + si2 + co3 + si3**

**}**

**else if (season == "dummy") {**

**D <- seasonaldummy(Yt)**

**D[rowSums(D) == 0, ] <- -1**

**smod <- Wt ~ -1 + D**

**}**

**else if (season == "none") {**

**}**

**else stop("Not a correct seasonal model is selected ('harmonic' or 'dummy') ")**

**Vt.bp <- 0**

**Wt.bp <- 0**

**CheckTimeTt <- 1**

**CheckTimeSt <- 1**

**i <- 0**

**while ((!identical(CheckTimeTt, Vt.bp) | !identical(CheckTimeSt,**

**Wt.bp)) & i < max.iter) {**

**CheckTimeTt <- Vt.bp**

**CheckTimeSt <- Wt.bp**

**Vt <- Yt - St**

**p.Vt <- sctest(efp(Vt ~ ti, h = h, type = "OLS-MOSUM"))**

**if (p.Vt$p.value <= 0.05) {**

**bp.Vt <- breakpoints(Vt ~ ti, h = h, breaks = breaks,**

**hpc = hpc)**

**nobp.Vt <- is.na(breakpoints(bp.Vt)[1])**

**}**

**else {**

**nobp.Vt <- TRUE**

**bp.Vt <- NA**

**}**

**if (nobp.Vt) {**

**fm0 <- rlm(Vt ~ ti)**

**Vt.bp <- 0**

**Tt <- ts(fitted(fm0))**

**tsp(Tt) <- tsp(Yt)**

**ci.Vt <- NA**

**}**

**else {**

**fm1 <- rlm(Vt ~ breakfactor(bp.Vt)/ti)**

**ci.Vt <- confint(bp.Vt, het.err = FALSE)**

**Vt.bp <- ci.Vt$confint[, 2]**

**Tt <- ts(fitted(fm1))**

**tsp(Tt) <- tsp(Yt)**

**}**

**if (season == "none") {**

**Wt <- 0**

**St <- 0**

**bp.Wt <- NA**

**ci.Wt <- NA**

**nobp.Wt <- TRUE**

**}**

**else {**

**Wt <- Yt - Tt**

**bp.Wt <- breakpoints(smod, h = h, breaks = breaks,**

**hpc = hpc)**

**nobp.Wt <- is.na(breakpoints(bp.Wt)[1])**

**if (nobp.Wt) {**

**sm0 <- rlm(smod)**

**St <- ts(fitted(sm0))**

**tsp(St) <- tsp(Yt)**

**Wt.bp <- 0**

**ci.Wt <- NA**

**}**

**else {**

**if (season == "dummy")**

**sm1 <- rlm(Wt ~ -1 + D %in% breakfactor(bp.Wt))**

**if (season == "harmonic")**

**sm1 <- rlm(Wt ~ (co + si + co2 + si2 + co3 +**

**si3) %in% breakfactor(bp.Wt))**

**St <- ts(fitted(sm1))**

**tsp(St) <- tsp(Yt)**

**ci.Wt <- confint(bp.Wt, het.err = FALSE)**

**Wt.bp <- ci.Wt$confint[, 2]**

**}**

**}**

**i <- i + 1**

**output[[i]] <- list(Tt = Tt, St = St, Nt = Yt - Tt -**

**St, Vt = Vt, bp.Vt = bp.Vt, Vt.bp = Vt.bp, ci.Vt = ci.Vt,**

**Wt = Wt, bp.Wt = bp.Wt, Wt.bp = Wt.bp, ci.Wt = ci.Wt)**

**}**

**if (!nobp.Vt) {**

**Vt.nrbp <- length(bp.Vt$breakpoints)**

**co <- coef(fm1)**

**Mag <- matrix(NA, Vt.nrbp, 3)**

**for (r in 1:Vt.nrbp) {**

**if (r == 1)**

**y1 <- co[1] + co[r + Vt.nrbp + 1] \* ti[Vt.bp[r]]**

**else y1 <- co[1] + co[r] + co[r + Vt.nrbp + 1] \***

**ti[Vt.bp[r]]**

**y2 <- (co[1] + co[r + 1]) + co[r + Vt.nrbp + 2] \***

**ti[Vt.bp[r] + 1]**

**Mag[r, 1] <- y1**

**Mag[r, 2] <- y2**

**Mag[r, 3] <- y2 - y1**

**}**

**index <- which.max(abs(Mag[, 3]))**

**m.x <- rep(Vt.bp[index], 2)**

**m.y <- c(Mag[index, 1], Mag[index, 2])**

**Magnitude <- Mag[index, 3]**

**Time <- Vt.bp[index]**

**}**

**else {**

**m.x <- NA**

**m.y <- NA**

**Magnitude <- 0**

**Time <- NA**

**Mag <- 0**

**}**

**return(structure(list(Yt = Yt, output = output, nobp = list(Vt = nobp.Vt,**

**Wt = nobp.Wt), Magnitude = Magnitude, Mags = Mag, Time = Time,**

**jump = list(x = ti[m.x], y = m.y)), class = "bfast"))**

**}**

**<environment: namespace:bfast>**