

Package ‘Rbeast’

November 24, 2021

Type Package

Version 0.9.1

Date 2021-11-22

Title Bayesian Change-Point Detection and Time Series Decomposition

Author Kaiguang Zhao [aut, cre],
Tongxi Hu [aut],
Yang Li [aut],
Jack Dongarra [ctb],
Cleve Moler [ctb]

Maintainer Kaiguang Zhao <lidar.rs@gmail.com>

Depends R (>= 2.10.0), methods, utils

Description Interpretation of time series data is affected by model choices. Different models can give different or even contradicting estimates of patterns, trends, and mechanisms for the same data--a limitation alleviated by the Bayesian estimator of abrupt change, seasonality, and trend (BEAST) of this package. BEAST seeks to improve time series decomposition by forgoing the "single-best-model" concept and embracing all competing models into the inference via a Bayesian model averaging scheme. It is a flexible tool to uncover abrupt changes (i.e., change-points), cyclic variations (e.g., seasonality), and nonlinear trends in time-series observations. BEAST not just tells when changes occur but also quantifies how likely the detected changes are true. It detects not just piecewise linear trends but also arbitrary nonlinear trends. BEAST is applicable to real-valued time series data of all kinds, be it for remote sensing, economics, climate sciences, ecology, and hydrology. Example applications include its use to identify regime shifts in ecological data, map forest disturbance and land degradation from satellite imagery, detect market trends in economic data, pinpoint anomaly and extreme events in climate data, and unravel system dynamics in biological data. Details on BEAST are reported in Zhao et al. (2019) <[doi:10.1016/j.rse.2019.04.034](https://doi.org/10.1016/j.rse.2019.04.034)>.

LazyData true

Imports grid

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-11-23 23:30:02 UTC

R topics documented:

beast	2
beast.irreg	15
beast123	23
CNAchrom11	41
geeLandsat	42
imagestack	44
minesweeper	45
ohio	46
plot.beast	47
print.beast	49
simdata	50
tetris	51
tsextract	53
Yellowstone	54

Index	56
--------------	-----------

beast	<i>Bayesian time series decomposition for changepoint, trend, and periodicity or seasonality</i>
-------	--

Description

A Bayesian model averaging algorithm called BEAST to decompose time series or 1D sequential data into individual components: Abrupt changes, trends, or periodic/seasonal variations. As a multi-purpose tool, it is useful for nonlinear trend analysis, time series decomposition, time series segmentation, changepoint detection, and detection of breakpoints or structural breaks.

Usage

```
beast(
  y,
  start      = 1,
  deltat     = 1,
  season     = c('harmonic','dummy','none'),
  freq       = NA,
  scp.minmax = c(0,10),  sorder.minmax = c(0,5),  sseg.min = NULL,
  tcp.minmax = c(0,10),  torder.minmax = c(0,1),  tseg.min = NULL,
  detrend    = FALSE,
  deseasonalize= FALSE,
  mcmc.seed  = 0,
  mcmc.burnin = 200,
  mcmc.chains = 3,
  mcmc.thin  = 5,
  mcmc.samples = 8000,
  print.options=TRUE,
```

```

    progressbar =TRUE,
    gui         = FALSE,
    ...
)

```

Arguments

- y** a vector for an evenly-spaced regular time series. Missing values such as NA and NaN are allowed.
- If *y* is irregular or unordered in time, use the `beast.irreg` function instead.
 - If *y* is quasi-regular (e.g., multiple years of daily data spanning across leap years: 365 points in some years, and 366 in others), also use the `beast.irreg` function instead.
 - If *y* is a matrix or 3D array consisting of multiple regular or irregular time series (e.g., stacked images), use `beast123` instead.
 - If *y* is an object of class 'ts', its time attributes (i.e., start, end, frequency) will be used to specify start, delta, and freq: their original values, if explicitly supplied to `beast`, will be overwritten. If frequency=1, season='none' is assumed regardless of its input value; if frequency>1 and if season='none' is specified by the user, 'none' will be replaced by 'harmonic'.
- start** numeric (default to 1.0) or Date; the time of the 1st datapoint of *y*. It can be specified as a scalar (e.g., 2021.0644), a vector of three values in the order of Year, Month, and Day (e.g., `c(2021, 1, 24)`), or a R's Date object (e.g., `as.Date('2021-1-24')`).
- deltat** numeric (default to 1.0); the time interval between consecutive data points. Its unit should be consistent with *start*. If *start* takes a numeric scalar, the unit is arbitrary and irrelevant to `beast` (e.g., 2021.3 can be of any unit). If *start* is a vector of Year, Month, and Day or an R's Date, *deltat* has the unit of YEAR. For example, if *start*=`c(2021, 1, 24)` for a monthly time series, *start* is converted to a fractional year $2021 + (24 - 0.5) / 365 = 2021.0644$ and *deltat*=1/12 needs to be set to specify the monthly interval.
- season** characters (default to 'harmonic'); specify if *y* has a periodic component or not. Three strings are possible.
- 'none': *y* is trend-only; no periodic components are present in the time series. The args for the seasonal component (i.e., `sorder.minmax`, `scp.minmax` and `sseg.max`) will be irrelevant and ignored.
 - 'harmonic': *y* has a periodic/seasonal component. The term *season* is a misnomer, being used here to broadly refer to any periodic variations present in *y*. The periodicity is NOT a model parameter estimated by BEAST but a known constant given by the user through *freq*. By default, the periodic component is modeled as a harmonic curve—a combination of sines and cosines.
 - If 'dummy', the same as 'harmonic' except that the periodic/seasonal component is modeled as a non-parametric curve. The harmonic order arg `sorder.minmax` is irrelevant and is ignored.

freq	numeric. Needed only for data with a periodic/cyclic component (i.e., season='harmonic' or 'dummy') and ignored for trend-only data (i.e., season='none'). It specifies the number of samples/datapoints per cycle (e.g, a time series of monthly observations with an annual period has a frequency of 12); it may be a decimal real number (e.g., a time series of bi-hourly observations with a period of 37.5 hrs has a freq of $37.5/2=18.75$). The period of the cyclic component in the unit of time is $\text{period}=\text{deltat}*\text{freq}$. freq is not a model parameter of BEAST and has to be specified by the user, but if freq is missing, BEAST first attempts to guess its value via auto-correlation before fitting the model: in this case, freq is assumed to be an integer.
sorder.minmax	a vector of 2 integers (≥ 1); the min and max harmonic orders considered to fit the seasonal component. sorder.minmax is used only if the time series has a seasonal component (i.e., season='harmonic') and ignored for trend-only data or when season='dummy'. If the min and max orders are equal (sorder.minmax[1]=sorder.minmax[2]), BEAST assumes a constant harmonic order used and won't infer the posterior probability of harmonic orders.
scp.minmax	a vector of 2 integers (≥ 0); the min and max number of seasonal changepts (scp) allowed in segmenting the seasonal component. scp.minmax is used only if y has a seasonal component (i.e., season='harmonic' or 'dummy') and ignored for trend-only data. If the min and max changepoint numbers are equal, BEAST assumes a constant number of scp and won't infer the posterior probability of the number of changepoints, but it still estimates the occurrence probability of the changepoints over time (i.e., the most likely times at which these changepoints occur). If both the min and max numbers are set to 0, no changepoints are allowed; then a global harmonic model is used to fit the seasonal component, but still, the most likely harmonic order will be inferred if sorder.minmax[1] is not equal to sorder.minmax[2].
sseg.min	an integer (> 0); the min segment length allowed between two neighboring season changepoints. That is, when fitting a piecewise harmonic seasonal model, no two changepoints are allowed to occur within a time window of length sseg.min. sseg.min must be an unitless integer—the number of time intervals/data points so that the time window in the original unit is $\text{sseg.min}*\text{deltat}$. sseg.min defaults to NULL and its value will be given a default value in reference to freq.
torder.minmax	a vector of 2 integers (≥ 0); the min and max orders of the polynomials considered to fit the trend component. The 0-th order corresponds to a constant term/a flat line and the 1st order is a line. If torder.minmax[1]=torder.minmax[2], BEAST assumes a constant polynomial order used and won't infer the posterior probability of polynomial orders.
tcp.minmax	a vector of 2 integers; the min and max number of trend changepoints (tcp) allowed in segmenting the trend component. If the min and max changepoint numbers are equal, BEAST assumes a constant number of changepoints and won't infer the posterior probability of the number of changepoints for the trend, but it still estimates the occurrence probability of the changepoints over time (i.e., the most likely times at which these changepoints occur in the trend). If both the min and max numbers are set to 0, no changepoints are allowed; then a global polynomial trend is used to fit the trend component, but still, the most likely polynomial order will be inferred if torder.minmax[1] is not equal to torder.minmax[2].

<code>tseg.min</code>	an integer (>0); the min segment length allowed between two neighboring trend changepoints. That is, when fitting a piecewise polynomial trend model, no two changepoints are allowed to occur within a time window of length <code>tseg.min</code> . <code>tseg.min</code> must be an unitless integer—the number of time intervals/data points so that the time window in the original unit is <code>tseg.min*delta.t</code> . <code>tseg.min</code> defaults to NULL and its value will be given a default value in reference to <code>freq</code> if the time series has a cyclic component.
<code>detrend</code>	logical; If TRUE, a global trend is first fitted and removed from the time series before running BEAST; after BEAST finishes, the global trend is added back to the BEAST result.
<code>deseasonalize</code>	logical; If TRUE, a global seasonal model is first fitted and removed from the time series before running BEAST; after BEAST finishes, the global seasonal curve is added back to the BEAST result. <code>deseasonalize</code> is ignored if <code>season='none'</code> (i.e., trend-only data).
<code>mcmc.seed</code>	integer (>=0); the seed for the random number generator used for Monte Carlo Markove Chain (mcmc). If <code>mcmc.seed=0</code> , an arbitrary seed is picked and the fitting results vary across runs. If fixed to the same non-zero integer, the result can be re-produced for different runs. But the results from the same seed may still vary if run on different computers because the random generator library depends on CPU's instruction sets.
<code>mcmc.chains</code>	integer (>0); the number of MCMC chains.
<code>mcmc.thin</code>	integer (>0); a factor to thin chains (e.g., if <code>thinningFactor=5</code> , samples will be taken every 3 iterations)
<code>mcmc.burnin</code>	integer (>0); the number of burn-in samples discarded at the start of each chain
<code>mcmc.samples</code>	integer (>=0); the number of samples collected per MCMC chain. The total number of iterations is $(\text{burnin} + \text{samples} * \text{thin}) * \text{chains}$.
<code>print.options</code>	boolean. If TRUE, the full list of input parameters to BEAST will be printed out prior to the MCMC inference; the naming for this list (e.g., metadata, prior, and mcmc) differs slightly from the input to <code>beast</code> , but there is a one-to-one correspondence (e.g., <code>prior\$trendMinSepDist=tseg.min</code>). Internally, <code>beast</code> converts the input parameters to the forms of metadata, prior, and mcmc. Type <code>'View(beast)'</code> to see the details or check the <code>beast123</code> function.
<code>progressbar</code>	boolean; control whether a progressbar will be displayed or not.
<code>gui</code>	boolean. If TRUE, BEAST will be run in a GUI demonstration mode, with a GUI window to show an animation of the MCMC sampling in the model space step by step. Note that "gui=TRUE" works only for Windows x64 systems not Windows 32 or Linux/Mac systems.
<code>...</code>	additional parameters. There are many more settings for the implementation but not made available in the <code>beast()</code> interface; please use the function <code>beast123()</code> instead

Value

The output is an object of class "beast". It is a list, consisting of the following variables. Its structure is the same as the outputs from the other two alternative functions `beast.irreg` and `beast123`. In the explanations below, we assume the input `y` is a single time series of length `N`:

time	a vector of size 1xN: the times at the N sampled locations. By default, it is simply set to 1:N
data	a vector, matrix, or 3D array; this is a copy of the input y if <code>extra\$dumpInputData=TRUE</code> . If <code>extra\$dumpInputData=FALSE</code> , it is set to NULL. If the original input y is irregular (as in <code>beast.irreg</code>), the copy here is the regular version aggregated from the original at the time interval specified by <code>deltat</code> (in <code>beast.irreg</code> or <code>metadata\$deltaTime</code> (in <code>beast123</code>)).
marg_lik	numeric; the average of the model marginal likelihood; the larger <code>marg_lik</code> , the better the fitting for a given time series.
R2	numeric; the R-square of the model fitting.
RMSE	numeric; the RMSE of the model fitting.
sig2	numeric; the estimated variance of the model error.
trend	a list object numeric consisting of various outputs related to the estimated trend component: <ul style="list-style-type: none"> • <code>ncp</code>: [Number of ChangePoints]. a numeric scalar; the mean number of trend changepoints. • <code>ncpPr</code>: [Probability of the Number of ChangePoints]. A vector of length $(\text{tcp.minmax}[2]+1)=\text{tcp.max}+1$. It gives a probability distribution of having a certain number of trend changepoints over the range of $[0,\text{tcp.max}]$; for example, <code>ncpPr[1]</code> is the probability of having no trend changepoint; <code>ncpPr[i]</code> is the probability of having (i-1) changepoints: Note that it is <code>ncpPr[i]</code> not <code>ncpPr[i-1]</code> because <code>ncpPr[1]</code> is used for having zero changepoint. • <code>cpOccPr</code>: [ChangePoint Occurrence Probability]. a vector of length N; it gives a probability distribution of having a changepoint in the trend at each point of time. Plotting <code>cpOccPr</code> will depict a continuous curve of probability-of-being-changepoint. <i>Of particular note, in the curve, a higher peak indicates a higher chance of being a changepoint only at that particular SINGLE point in time and does not necessarily mean a higher chance of observing a changepoint AROUND that time. For example, a window of <code>cpOccPr</code> values $c(0,0,0.5,0,0)$ (i.e., the peak prob is 0.5 and the summed prob is 0.5) is less likely to be a changepoint compared to another window $c(0.1,0.2,0.21,0.2,0.1)$ (i.e., the peak prob is 0.21 but the summed prob is 0.71).</i> • <code>order</code>: a vector of length N; the average polynomial order needed to approximate the fitted trend. As an average over many sampled individual piece-wise polynomial trends, <code>order</code> is not necessarily an integer. • <code>cp</code>: [Changepoints] a vector of length <code>tcp.max=tcp.minmax[2]</code>; the most possible changepoint locations in the trend component. The locations are obtained by first applying a sum-filtering to the <code>cpOccPr</code> curve with a filter window size of <code>tseg.min</code> and then picking up to a total <code>ncp</code> of the highest peaks in the filtered curve. If <code>ncp<tcp.max</code>, the remaining of the vector is filled with NaNs. • <code>cpPr</code>: [Changepoints Probability] a vector of length <code>tcp.max=tcp.minmax[2]</code>; the probabilities associated with the changepoints <code>cp</code>. Filled with NaNs for the remaining elements if <code>ncp<tcp.max</code>.

- cpCI: [Changepoints Credible Interval] a matrix of dimension $\text{tcp.max} \times 2$; the credible intervals for the detected changepoints cp.
- cpAbruptChange: [Abrupt change at Changepoints] a vector of length tcp.max ; the jumps in the fitted trend curves at the detected changepoints cp.
- Y: a vector of length N; the estimated trend component. It is the Bayesian model averaging of all the individual sampled trend.
- SD: [Standard Deviation] a vector of length N; the estimated standard deviation of the estimated trend component.
- CI: [Standard Deviation] a matrix of dimension $N \times 2$; the estimated credible interval of the estimated trend. One vector of the matrix is for the upper envelope and another for the lower envelope.
- slp: [Slope] a vector of length N; the time-varying slope of the fitted trend component .
- slpSD: [Standar Deviation of Slope] a vector of length N; the SD of the slope for the trend component.
- slpSignPr: [PRobability of slope having a postive sign] a vector of length N; the probability of the slope being positive (i.e., increasing trend) for the trend component. For example, if $\text{slpSignPr}=0.80$ at a given point in time, it means that 80
- pos_ncp:
- neg_ncp:
- pos_ncpPr:
- neg_ncpPr:
- pos_cpOccPr:
- neg_cpOccPr:
- pos_cp:
- neg_cp:
- pos_cpPr:
- neg_cpPr:
- pos_cpAbruptChange:
- neg_cpAbruptChange:
- pos_cpCI:
- neg_cpCI: The above variables have the same outputs as those variables without the prefix 'pos' and 'neg', except that we differentiate the changepoints with a POSitive jump in the trend from those changepoints with a NEGative jump. For example, pos_ncp refers to the average number of trend changepoints that jump up (i.e., postively) in the trend.
- inc_ncp:
- dec_ncp:
- inc_ncpPr:
- dec_ncpPr:
- inc_cpOccPr:
- dec_cpOccPr:
- inc_cp:

- `dec_cp`:
- `inc_cpPr`:
- `dec_cpPr`:
- `inc_cpAbruptChange`:
- `dec_cpAbruptChange`:
- `inc_cpCI`:
- `dec_cpCI`: The above variables have the same outputs as those variables without the prefix 'inc' and 'dec', except that we differentiate the change-points at which the trend slope increases from those changepoints at which the trend slope decreases. For example, if the trend slopes before and after a `chnp` is 0.4 and 2.5, then the changepoint is counted toward `inc_ncp`.

season

a list object numeric consisting of various outputs related to the estimated seasonal/periodic component:

- `ncp`: [Number of ChangePoints]. a numeric scalar; the mean number of seasonal changepoints.
- `ncpPr`: [Probability of the Number of ChangePoints]. A vector of length $(\text{scp.minmax}[2]+1)=\text{scp.max}+1$. It gives a probability distribution of having a certain number of seasonal changepoints over the range of $[0,\text{scp.max}]$; for example, `ncpPr[1]` is the probability of having no seasonal change-point; `ncpPr[i]` is the probability of having (i-1) changepoints: Note that the index is i rather than (i-1) because `ncpPr[1]` is used for having zero changepoint.
- `cpOccPr`: [ChangePoint OCCurrence PRObability]. a vector of length N; it gives a probability distribution of having a changepoint in the seasonal component at each point of time. Plotting `cpOccPr` will depict a continuous curve of probability-of-being-changepoint over the time. Of particular note, in the curve, a higher value at a peak indicates a higher chance of being a changepoint only at that particular SINGLE point in time, and does not necessarily mean a higher chance of observing a changepoint AROUND that time. For example, a window of `cpOccPr` values $c(0, 0, 0.5, 0, 0)$ (i.e., the peak prob is 0.5 and the summed prob is 0.5) is less likely to be a change-point compared to another window values $c(0.1, 0.2, 0.3, 0.2, 0.1)$ (i.e., the peak prob is 0.3 but the summed prob is 0.8).
- `order`: a vector of length N; the average harmonic order needed to approximate the seasonal component. As an average over many sampled individual piece-wise harmonic curves, `order` is not necessarily an integer.
- `cp`: [Changepoints] a vector of length $\text{scp.max}=\text{scp.minmax}[2]$; the most possible changepoint locations in the seasonal component. The locations are obtained by first applying a sum-filtering to the `cpOccPr` curve with a filter window size of `sseg.min` and then picking up to a total `ncp` of the highest peaks in the filtered curve. If `ncp<scp.max`, the remaining of the vector is filled with NaNs.
- `cpPr`: [Changepoints PRObability] a vector of length `scp.max`; the probabilities associated with the changepoints `cp`. Filled with NaNs for the remaining elements if `ncp<scp.max`.

- cpCI: [Changepoints Credible Interval] a matrix of dimension $\text{scp.max} \times 2$; the credible intervals for the detected changepoints cp.
- cpAbruptChange: [Abrupt change at Changepoints] a vector of length scp.max ; the jumps in the fitted trend curves at the detected changepoints cp.
- Y: a vector of length N; the estimated trend component. It is the Bayesian model averaging of all the individual sampled trend.
- SD: [Standard Deviation] a vector of length N; the estimated standard deviation of the estimated trend component.
- CI: [Standard Deviation] a matrix of dimension $N \times 2$; the estimated credible interval of the estimated trend. One vector of the matrix is for the upper envelope and another for the lower envelope.
- amp: [AMPLitude] a vector of length N; the time-varying amplitude of the estimated seasonality.
- ampSD: [Standar Deviation of AMPLitude] a vector of length N; , the SD of the amplitude of the seasonality.
- pos_ncp:
- neg_ncp:
- pos_ncpPr:
- neg_ncpPr:
- pos_cpOccPr:
- neg_cpOccPr:
- pos_cp:
- neg_cp:
- pos_cpPr:
- neg_cpPr:
- pos_cpAbruptChange:
- neg_cpAbruptChange:
- pos_cpCI:
- neg_cpCI: The above variables have the same outputs as those variables without the prefix 'pos' and 'neg', except that we differentiate the change-points with a POSitive jump in the trend from those changepoints with a NEGative jump. For example, pos_ncp refers to the average number of trend changepoints that jump up (i.e., postively) in the trend.

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).

- Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. ISPRS Journal of Photogrammetry and Remote Sensing, 176, pp.250-261(a beast application paper).

See Also

[beast](#), [beast.irreg](#), [beast123](#), [minesweeper](#), [tetris](#), [geeLandsat](#)

Examples

```
library(Rbeast)
```

```
#####
# Yellowstone is a half-monthly satellite time series of 774 NDVI(vegetation greeness)
# observations starting from July 1-15,1981(i.e., start=c(1981,7,7)) at a Yellowstone
# site. It has 24 data points per year (i.e., freq=24). Note that the beast function
# handles only evenly-spaced regular time series. Irregular data need to be first
# aggregated at a regular time interval of your choice--the aggregation functionality
# is implemented in beast.irreg() and beast123().
```

```
data(Yellowstone)
plot(Yellowstone)
```

```
#####Example 1#####
# Yellowstone is not a object of class 'ts' but a pure vector without time attributes.
# Below, no extra argument is supplied so default values (i.e.,start=1, deltat=1) are
# used and the time is 1:774. 'freq' is missing and so is guessed via auto-correlation.
# Use of auto-correlation to compute the period of a cyclic time series is not always
# reliable, so it is suggested to always supply 'freq' directly, as in Example 2 and
# Example 3.
```

```
out = beast(Yellowstone) # the times assumed to be 1:length(Yellowstone) by default
plot(out)
```

```
#####Example 2#####
# The time info (e.g., start,delta,and freq) are explicitly provided. 'start' can be
# given as a vector comprising year, month,& day, a fractional number, or a R's Date.
# The unit of start and deltat does not necessarily refer to time and can be arbitrary
# (e.g., a sequence of data observed at evenly-spaced distaces along a transect or a
# elevation gradient)
```

```
o=beast(Yellowstone, start = c(1981,7,7),          deltat=1/24, freq=24)
#o=beast(Yellowstone, start = 1981.5137,          deltat=1/24, freq=24)
#o=beast(Yellowstone, start = as.Date('1981-7-7'), deltat=1/24, freq=24)
```

```
print(o) # o is a R LIST object with many fields
plot(o) # plot many variables
plot(o, vars=c('st','s','t')) # plot the Y, seasonal, and trend components only
plot(o, vars=c('s','scp','samp','t','tcp','tslp'))# plot some selected variables in
```

```

# 'o'. Type "?plot.beast" to see
# more about vars
plot(o, vars=c('s','t'),col=c('red','blue')) # specify colors of selected subplots

plot(o$time, o$season$Y,type='l') # directly plot output: the fitted season
plot(o$time, o$season$cpOccPr[,1])# directly plot output: season chgpt prob
plot(o$time, o$trend$Y,type='l') # directly plot output: the fitted trend
plot(o$time, o$trend$cpOccPr[,1])# directly plot output: trend chgpt occurrence prob
plot(o$time, o$season$order[,1]) # directly plot output: avg harmonic order

plot(o, interactive=TRUE) # manually choose which variables to plot

#####Example 3#####
# Specify other arguments explicitly. Default values are used for missing parameters.
# Note that beast() and beast123() call the same internal C/C++ library, so in beast,
# the input parameters will be converted to metadata, prior, mcmc, and extra parameters
# as explained for the beast123() function.

out = beast(Yellowstone, freq=24,season='dummy', mcmc.samples=5000, tseg.min=20)
plot(out)

out=beast(
  Yellowstone, # Yellowstone: a pure numeric vector wo time info
  start=1981.51, deltat=1/24, freq = 24,
  season='harmonic', # periodic compnt exists,fitted as a harmonic curve
  scp.minmax = c(0,3), # min and max numbers of seasonal changpts allowed
  sorder.minmax = c(1,5), # min and max harmonic orders allowed
  sseg.min = 24, # the min length of segments btw neighboring chnpts
  tcp.minmax = c(0,10),# min and max numbers of changpts allowed in the trend
  torder.minmax = c(0,1), # min and maxx polynomial orders to fit trend
  tseg.min = 24, # the min length of segments btw neighboring trend chnpts
  deseasonalize = TRUE, # remove the global seasonality before fitting the beast model
  detrend = TRUE, # remove the global trend before fitting the beast model
  mcmc.seed = 0, # a seed for mcmc's random nummber generator; use a
  # non-zero integer to reproduce results across runs
  mcmc.burnin = 500, # number of initial iterations discarded
  mcmc.chains = 2, # number of chains
  mcmc.thin = 3, # include samples every 3 iterations
  mcmc.samples = 6000 # number of samples taken per chain
  # total iteration: (500+3*6000)*2
)
plot(out)
plot(out, interactive=TRUE)

#####Example 4#####
# Run an interactive GUI to visualize how BEAST is samplinig from the possible model
# spaces in terms of the numbers and timings of seasonal and trend changepoints.
# The GUI inferface allows changing the option parameters interactively. This GUI is

```

```
# only available on Win x64 machines, not Mac or Linux.
```

```
beast(Yellowstone, freq=24, gui=TRUE)
```

```
#####
# Apply beast to trend-only data. 'Nile' is the ANNUAL river flow of the river
# Nile at Aswan since 1871. It is a 'ts' object; its time attributes (start=1871,
# end=1970,frequency=1) are used to replace the user-supplied start,deltat, and freq,
# if any.
```

```
data(Nile)
plot(Nile)
attributes(Nile) # a ts object with time attributes (i.e., tsp=(start,end,freq)
```

```
o=beast(Nile)          # start=1871, delta=1, and freq=1 taken from Nile itself
plot(o)
```

```
o=beast(Nile,          # the same as above. The user-supplied values are igored
      start=2011,      # bcz Nile has its own time attributes. Its frequency tag
      freq=24,         # is 1, so season='none'/trend-only is assumed instead of
      season='harmonic') # the supplied 'harmonic'
```

```
#####Example 5#####
# NileVec is a pure data vector. The first run below is WRONG bcz NileVec was assumed
# to have a perodic component by default and beast gets a best estimate of freq=6 while
# the true value is freq=1. To fit a trend-only model, season='none' has to be explicitly
# specified, as in the 2nd & 3rd funs.
```

```
NileVec = as.vector(Nile) # NileVec is not a ts obj but a pure numeric data vector
o = beast(NileVec)        # ERROR: No time attributes available to interpret NileVec.
                          # By default, beast assumes season='harmonic', start=1,
                          # and deltat=1. 'freq' is guessed to be 6 (WRONG).
plot(o)
```

```
o=beast(NileVec,season='none') # Use season='none' for trend-only analysis; the default
                              # time is the indices '1:length(NileVec)'.
plot(o)
```

```
o=beast(Nile,          # The true time attributes, if needed, has to be given
      start = 1871,    # explicitly thru start and deltat (or freq if there is
      deltat = 1,      # a cyclic cmpnt).
      season = 'none')
plot(o)
```

```
#####Example 6#####
# beast can handle missing data. co2 is a monthly time series (i.e.,freq=12) starting
# from Jan 1959. We generate some missing values at random indices
```

```

data(co2)
attributes(co2)           # A ts object with time attributes (i.e., tsp)
badIdx    = sample( 1:length(co2), 50) # Get a set of random indices
co2[badIdx] = NA           # Insert some data gaps

out=beast(co2) # co2 is a ts object and its 'tsp' time attributes are used to get the
               # true time info. No need to specify 'start','deltat', & freq explicitly.

out=beast(co2,           # the supplied time/freq values will be ignored bcz
           start=c(1959,1,15), # co2 is a ts object; the correct freq = 12 will be
           deltat=1/12,       # used
           freq=365)
print(out)
plot(out)

```

```

#####Example 7#####
# Apply beast to time series-like sequence data: the unit of sequences is not
# necessarily time.

```

```

data(CNACHrom11) # DNC copy number alterations in Chromosome 11 for cell line GM05296
                 # The data is ordered by genomic position (not time), and the values
                 # are the log2-based intensity ratio of copy numbers between the sample
                 # the reference. A value of zero means no gain or loss in copy number.
o = beast(CNACHrom11,season='none') # season is a misnomer here bcz the data has nothing
                                   # to do with time. Regardless, we fit only a trend.
plot(o)

```

```

#####Example 8#####
# Apply beast to time series-like data: the unit of sequences is not necessarily time.

```

```

# Age of Death of Successive Kings of England
# If the data link is deprecated, install the time series data library instead,
# which is available at https://pkg.yangzhuoranyang.com/tsdl/
# install.packages("devtools")
# devtools::install_github("FinYang/tsdl")
# kings = tsdl::tsdl[[293]]

kings = scan("http://robjhyndman.com/tsdldata/misc/kings.dat",skip=3)
out    = beast(kings,season='none')
plot(out)

```

```
#####Example 9#####
# Another example from the tsdl data library
```

```
# Number of monthly births in New York from Jan 1946 to Dec 1959
# If the data link becomes invalid, install the time series data package instead
# install.packages("devtools")
# devtools::install_github("FinYang/tsdl")
# kings = tsdl::tsdl[[534]]

births = scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")
out = beast(births,start=c(1946,1,15), deltat=1/12 )
plot(out) # the result is wrong bcz the guessed freq via auto-correlation by beast
          # is 2 rather than 12, so we recommend always specifying 'freq' explicitly
          # for those time series with a periodic component, as shown below.
out = beast(births,start=c(1946,1,15), deltat=1/12, freq=12 )
plot(out)
```

```
#####Example10#####
# The old API interface of beast is still made available but not recommended. It is
# kept mainly to ensure the working of the sample code on Page 475 in the text
# Ecological Methods by Drs. Southwood and Henderson.
```

```
# The interface as shown here will be deprecated and not recommended.
beast(Yellowstone, 24) #24 is the freq: number of datapoints per period
```

```
# Specify the model or MCMC parameters through opt as in Rbeast v0.2
opt=list() #Create an empty list to append individual model parameters
opt$period=24 #Period of the cyclic component (i.e.,freq in the new version)
opt$minSeasonOrder=2 #Min harmonic order allowed in fitting season component
opt$maxSeasonOrder=8 #Max harmonic order allowed in fitting season component
opt$minTrendOrder=0 #Min polynomial order allowed to fit trend (0 for constant)
opt$maxTrendOrder=1 #Max polynomial order allowed to fit trend (1 for linear term)
opt$minSepDist_Season=20#Min separation time btw neighboring season changepoints
opt$minSepDist_Trend=20 #Min separation time btw neighboring trend changepoints
opt$maxKnotNum_Season=4 #Max number of season changepoints allowed
opt$maxKnotNum_Trend=10 #Max number of trend changepoints allowed
opt$omittedValue=NA #A customized value to indicate bad/missing values in the time
#series, in addition to those NA or NaN values.
```

```
# The following parameters used to configure the reversible-jump MCMC (RJMCC) sampler
opt$chainNumber=2 #Number of parallel MCMC chains
opt$sample=1000 #Number of samples to be collected per chain
opt$thinningFactor=3 #A factor to thin chains
opt$burnin=500 #Number of burn-in samples discarded at the start
opt$maxMoveStepSize=30 #For the move proposal, the max window allowed in jumping from
```

```

#the current changepoint
opt$resamplingSeasonOrderProb=0.2 #The probability of selecting a re-sampling proposal
#(e.g., resample seasonal harmonic order)
opt$resamplingTrendOrderProb=0.2 #The probability of selecting a re-sampling proposal
#(e.g., resample trend polynomial order)

opt$seed=65654 #A seed for the random generator: If seed=0,random numbers differ
#for different BEAST runs. Setting seed to a chosen non-zero integer
#will allow reproducing the same result for different BEAST runs.

beast(Yellowstone, opt)

```

beast.irreg	<i>Bayesian time series decomposition for changepoint, trend, and periodicity or seasonality</i>
-------------	--

Description

A Bayesian model averaging algorithm called BEAST to decompose time series or 1D sequential data into individual components: Abrupt changes, trends, or periodic/seasonal variations. As a multi-purpose tool, it is useful for nonlinear trend analysis, time series decomposition, time series segmentation, changepoint detection, and detection of breakpoints or structural breaks.

Usage

```

beast.irreg(
  y,
  time, deltat,
  freq      = NA,
  season    = c('harmonic', 'dummy', 'none'),
  scp.minmax = c(0,10),  sorder.minmax = c(0,5),  sseg.min = NULL,
  tcp.minmax = c(0,10),  torder.minmax = c(0,1),  tseg.min = NULL,
  detrend    = FALSE,
  deseasonalize= FALSE,
  mcmc.seed  = 0,
  mcmc.burnin = 200,
  mcmc.chains = 3,
  mcmc.thin  = 5,
  mcmc.samples = 8000,
  print.options= TRUE,
  progressbar = TRUE,
  gui         = FALSE,
  ...
)

```

Arguments

<code>y</code>	<p>a vector for an irregular or unordered time series. Missing values such as NA and NaN are allowed.</p> <ul style="list-style-type: none"> • If <code>y</code> is regular (i.e., evenly-spaced in time), use the <code>beast</code> function instead. • If <code>y</code> is a matrix or 3D array consisting of multiple regular or irregular time series (e.g., stacked images), use <code>beast123</code> instead.
<code>time</code>	<p>a vector of the same length as <code>y</code>; the time vector of datapoints of <code>y</code>. It can be a numeric vector or a vector of Dates. If your time is formatted as strings or vectors of year, months, and days, use <code>beast123()</code> instead where more formats are supported.</p>
<code>deltat</code>	<p>numeric; a user-specified time interval to aggregate <code>y</code> based on <code>time</code> into a regular time series. The BEAST model is currently formulated for regular data only, so internally, the <code>beast.irreg</code> function will aggregate/re-bin irregular data into regular ones; for the aggregation, the <code>deltat</code> MUST be provided to specify the desired bin size or time interval. The unit of delta needs to be consistent with <code>time</code>. If <code>time</code> takes a numeric vector, the unit is arbitrary and irrelevant to <code>beast</code>. If <code>time</code> takes a vector of Dates, the unit is assumed to YEAR; for example, if the desired time interval is 1 month (or 1 day), <code>deltat</code> should be 1/12 (or 1/365).</p>
<code>freq</code>	<p>numeric. Needed only for data with a periodic/cyclic component (i.e., <code>season='harmonic'</code> or <code>'dummy'</code>) and ignored for trend-only data (i.e., <code>season='none'</code>). It specifies the number of samples/datapoints per cycle (e.g., a time series of monthly observations with an annual period has a frequency of 12); it may be a decimal real number (e.g., a time series of bi-hourly observations with a period of 37.5 hrs has a <code>freq</code> of $37.5/2=18.75$). The period of the cyclic component in the unit of time is <code>period=deltat*freq</code>. <code>freq</code> is not a model parameter of BEAST and it has to be specified by the user. If <code>freq</code> is missing, BEAST first attempts to guess its value via auto-correlation before fitting the model, but in this case, <code>freq</code> is assumed to be an integer.</p>
<code>season</code>	<p>characters (default to <code>'harmonic'</code>); specify if <code>y</code> has a periodic component or not. Three strings are possible.</p> <ul style="list-style-type: none"> • <code>'none'</code>: <code>y</code> is trend-only; no periodic components are present in the time series. The args for the seasonal component (i.e., <code>sorder.minmax</code>, <code>scp.minmax</code> and <code>sseg.max</code>) will be ignored. • <code>'harmonic'</code>: <code>y</code> has a periodic/seasonal component. The term <code>'season'</code> is a misnomer, being used here to broadly refer to any periodic variations present in <code>y</code>. The periodicity is NOT a model parameter estimated by <code>beast</code> but a known constant given by the user through <code>freq</code>. By default, the periodic component is modeled as a harmonic curve—a combination of sines and cosines. • If <code>'dummy'</code>, the same as <code>'harmonic'</code> except that the periodic/seasonal component is modeled as a non-parametric curve. The arg <code>sorder.minmax</code> is irrelevant and is ignored.
<code>sorder.minmax</code>	<p>a vector of 2 integers (≥ 1); the min and max harmonic orders considered to fit the seasonal component. <code>sorder.minmax</code> is used only used if the time series has a seasonal component (i.e., <code>season='harmonic'</code>) and ignored for trend-only data or when <code>season='dummy'</code>. If the min and max orders are equal</p>

	(<code>sorder.minmax[1]=sorder.minmax[2]</code>), BEAST assumes a constant harmonic order used and won't infer the posterior probability of harmonic orders.
<code>scp.minmax</code>	a vector of 2 integers (≥ 0); the min and max number of seasonal changepts (scp) allowed in segmenting the seasonal component. <code>scp.minmax</code> is used only if <code>y</code> has a seasonal component (i.e., <code>season='harmonic'</code> or <code>'dummy'</code>) and ignored for trend-only data. If the min and max changepoint numbers are equal, BEAST assumes a constant number of scp and won't infer the posterior probability of the number of changepts, but it still estimates the occurrence probability of the changepts over time (i.e., the most likely times at which these changepts occur). If both the min and max numbers are set to 0, no changepts are allowed; then a global harmonic model is used to fit the seasonal component, but still, the most likely harmonic order will be inferred if <code>sorder.minmax[1]</code> is not equal to <code>sorder.minmax[2]</code> .
<code>sseg.min</code>	an integer; the min segment length allowed between two neighboring season changepts. That is, when fitting a piecewise harmonic seasonal model, no two changepts are allowed to occur within a time window of length <code>sseg.min</code> . <code>sseg.min</code> must be an unitless integer—the number of time intervals/data points so that the time window in the original unit is <code>sseg.min*deltat</code> . <code>sseg.min</code> defaults to NULL and its value will be given a default value in reference to <code>freq</code> .
<code>torder.minmax</code>	a vector of 2 integers (≥ 0); the min and max orders of the polynomials considered to fit the trend component. The 0-th order corresponds to a constant term/a flat line and the 1st order is a line. If <code>torder.minmax[1]=torder.minmax[2]</code> , BEAST assumes a constant polynomial order used and won't infer the posterior probability of polynomial orders.
<code>tcp.minmax</code>	a vector of 2 integers; the min and max number of trend changepts (tcp) allowed in segmenting the trend component. If the min and max changepoint numbers are equal, BEAST assumes a constant number of changepts and won't infer the posterior probability of the number of changepts for the trend, but it still estimates the occurrence probability of the changepts over time (i.e., the most likely times at which these changepts occur in the trend). If both the min and max numbers are set to 0, no changepts are allowed; then a global polynomial trend is used to fit the trend component, but still, the most likely polynomial order will be inferred if <code>torder.minmax[1]</code> is not equal to <code>torder.minmax[2]</code> .
<code>tseg.min</code>	an integer; the min segment length allowed between two neighboring trend changepts. That is, when fitting a piecewise polynomial trend model, no two changepts are allowed to occur within a time window of length <code>tseg.min</code> . <code>tseg.min</code> must be an unitless integer—the number of time intervals/data points so that the time window in the original unit is <code>tseg.min*deltat</code> . <code>tseg.min</code> defaults to NULL and its value will be given a default value in reference to <code>freq</code> if the time series has a cyclic component.
<code>detrend</code>	logical; If TRUE, a global trend is first fitted and removed from the time series before running BEAST; after BEAST finishes, the global trend is added back to the BEAST result.
<code>deseasonalize</code>	logical; If TRUE, a global seasonal model is first fitted and removed from the time series before running BEAST; after BEAST finishes, the global seasonal curve is added back to the BEAST result. <code>deseasonalize</code> is ignored if <code>season='none'</code> (i.e., trend-only data).

mcmc.seed	integer (≥ 0); the seed for the random number generator used for Monte Carlo Markove Chain (mcmc). If <code>mcmc.seed=0</code> , an arbitrary seed is picked and the fitting results vary across runs. If fixed to the same non-zero integer, the result can be re-produced for different runs. But the results from the same seed may still vary if run on different computers because the random generator library depends on CPU's instruction sets.
mcmc.chains	integer (> 0); the number of MCMC chains.
mcmc.thin	integer (> 0); a factor to thin chains (e.g., if <code>thinningFactor=5</code> , samples will be taken every 5 iterations)
mcmc.burnin	integer (> 0); the number of burn-in samples discarded at the start of each chain
mcmc.samples	integer (≥ 0); the number of samples collected per MCMC chain. The total number of iterations is $(\text{burnin} + \text{samples} * \text{thin}) * \text{chains}$.
print.options	boolean. If TRUE, the full list of input parameters to BEAST will be printed out prior to the MCMC inference; the naming for this list (e.g., metadata, prior, and mcmc) differs slightly from the input to <code>beast</code> , but there is a one-to-one correspondence (e.g., <code>prior\$trendMinSepDist=tseg.min</code>). Internally, <code>beast</code> converts the input parameters to the forms of metadata, prior, and mcmc. Type <code>'View(beast)'</code> to see the details or check the <code>beast123</code> function.
progressbar	boolean; control whether a progressbar will be displayed or not.
gui	boolean. If TRUE, BEAST will be run in a GUI demonstration mode, with a GUI window to show an animation of the MCMC sampling in the model space step by step. Note that "gui=TRUE" works only for Windows x64 systems not Windows 32 or Linux/Mac systems.
...	additional parameters. There are many more settings for the implementation but not made available in the <code>beast()</code> interface; please use the function <code>beast123()</code> instead

Value

The output is an object of class "beast". It is a list, consisting of the following variables. In the explanations below, we assume the input `y` is a single time series of length `N`:

time	a vector of size $1 \times N$: the times at the <code>N</code> sampled locations. By default, it is simply set to $1:N$
data	a vector, matrix, or 3D array; this is a copy of the input <code>data</code> if <code>extra\$dumpInputData = TRUE</code> . If <code>extra\$dumpInputData=FALSE</code> , it is set to NULL. If the original input data is irregular, the copy here is the regular version aggregated from the original at the time interval specified by <code>metadata\$deltaTime</code> .
marg_lik	numeric; the average of the model marginal likelihood; the larger <code>marg_lik</code> , the better the fitting for a given time series.
R2	numeric; the R-square of the model fitting.
RMSE	numeric; the RMSE of the model fitting.
sig2	numeric; the estimated variance of the model error.
trend	a list object numeric consisting of various outputs related to the estimated trend component:

- `ncp`: [Number of ChangePoints]. a numeric scalar; the mean number of trend changepoints.
- `ncpPr`: [Probability of the Number of ChangePoints]. A vector of length $(\text{tcp.minmax}[2]+1)=\text{tcp.max}+1$. It gives a probability distribution of having a certain number of trend changepoints over the range of $[0,\text{tcp.max}]$; for example, `ncpPr[1]` is the probability of having no trend changepoint; `ncpPr[i]` is the probability of having (i-1) changepoints: Note that it is `ncpPr[i]` not `ncpPr[i-1]` because `ncpPr[1]` is used for having zero changepoint.
- `cpOccPr`: [ChangePoint OCCurrence PRObability]. a vector of length N; it gives a probability distribution of having a changepoint in the trend at each point of time. Plotting `cpOccPr` will depict a continuous curve of probability-of-being-changepoint. Of particular note, in the curve, a higher peak indicates a higher chance of being a changepoint only at that particular SINGLE point in time and does not necessarily mean a higher chance of observing a changepoint AROUND that time. For example, a window of `cpOccPr` values $c(0, 0, 0.5, 0, 0)$ (i.e., the peak prob is 0.5 and the summed prob is 0.5) is less likely to be a changepoint compared to another window $c(0.1, 0.2, 0.21, 0.2, 0.1)$ (i.e., the peak prob is 0.21 but the summed prob is 0.71).
- `order`: a vector of length N; the average polynomial order needed to approximate the fitted trend. As an average over many sampled individual piece-wise polynomial trends, `order` is not necessarily an integer.
- `cp`: [Changepoints] a vector of length $\text{tcp.max}=\text{tcp.minmax}[2]$; the most possible changepoint locations in the trend component. The locations are obtained by first applying a sum-filtering to the `cpOccPr` curve with a filter window size of `tseg.min` and then picking up to a total `ncp` of the highest peaks in the filtered curve. If `ncp<tcp.max`, the remaining of the vector is filled with NaNs.
- `cpPr`: [Changepoints PRObability] a vector of length $\text{tcp.max}=\text{tcp.minmax}[2]$; the probabilities associated with the changepoints `cp`. Filled with NaNs for the remaining elements if `ncp<tcp.max`.
- `cpCI`: [Changepoints Credible Interval] a matrix of dimension $\text{tcp.max} \times 2$; the credible intervals for the detected changepoints `cp`.
- `cpAbruptChange`: [Abrupt change at Changepoints] a vector of length tcp.max ; the jumps in the fitted trend curves at the detected changepoints `cp`.
- `Y`: a vector of length N; the estimated trend component. It is the Bayesian model averaging of all the individual sampled trend.
- `SD`: [Standard Deviation] a vector of length N; the estimated standard deviation of the estimated trend component.
- `CI`: [Standard Deviation] a matrix of dimension $N \times 2$; the estimated credible interval of the estimated trend. One vector of the matrix is for the upper envelope and another for the lower envelope.
- `s1p`: [Slope] a vector of length N; the time-varying slope of the fitted trend component.
- `s1pSD`: [Standard Deviation of Slope] a vector of length N; the SD of the slope for the trend component.

- `slpSignPr`: [PProbability of slope having a postive sign] a vector of length `N`; the probability of the slope being positive (i.e., increasing trend) for the trend component. For example, if `slpSignPr=0.80` at a given point in time, it means that 80
- `pos_ncp`:
- `neg_ncp`:
- `pos_ncpPr`:
- `neg_ncpPr`:
- `pos_cpOccPr`:
- `neg_cpOccPr`:
- `pos_cp`:
- `neg_cp`:
- `pos_cpPr`:
- `neg_cpPr`:
- `pos_cpAbruptChange`:
- `neg_cpAbruptChange`:
- `pos_cpCI`:
- `neg_cpCI`: The above variables have the same outputs as those variables without the prefix 'pos' and 'neg', except that we differentiate the change-points with a POSitive jump in the trend from those changepoints with a NEGative jump. For example, `pos_ncp` refers to the average number of trend changepoints that jump up (i.e., postively) in the trend.
- `inc_ncp`:
- `dec_ncp`:
- `inc_ncpPr`:
- `dec_ncpPr`:
- `inc_cpOccPr`:
- `dec_cpOccPr`:
- `inc_cp`:
- `dec_cp`:
- `inc_cpPr`:
- `dec_cpPr`:
- `inc_cpAbruptChange`:
- `dec_cpAbruptChange`:
- `inc_cpCI`:
- `dec_cpCI`: The above variables have the same outputs as those variables without the prefix 'inc' and 'dec', except that we differentiate the change-points at which the trend slope increases from those changepoints at which the trend slope decreaes. For example, if the trend slopes before and after a chngpt is 0.4 and 2.5, then the changepont is counted toward `inc_ncp`.

`season`

a list object numeric consisting of various outputs related to the estimated seasonal/periodic component:

- `ncp`: [Number of ChangePoints]. a numeric scalar; the mean number of seasonal changepoints.

- `ncpPr`: [Probability of the Number of ChangePoints]. A vector of length $(\text{scp.minmax}[2]+1)=\text{scp.max}+1$. It gives a probability distribution of having a certain number of seasonal changepoints over the range of $[0,\text{scp.max}]$; for example, `ncpPr[1]` is the probability of having no seasonal changepoint; `ncpPr[i]` is the probability of having (i-1) changepoints: Note that the index is *i* rather than (i-1) because `ncpPr[1]` is used for having zero changepoint.
- `cpOccPr`: [ChangePoint OCCurrence PRobability]. a vector of length *N*; it gives a probability distribution of having a changepoint in the seasonal component at each point of time. Plotting `cpOccPr` will depict a continous curve of probability-of-being-changepoint over the time. Of particular note, in the curve, a higher value at a peak indicates a higher chance of being a changepoint only at that particular SINGLE point in time, and does not necessarily mean a higher chance of observing a changepoint AROUND that time. For example, a window of `cpOccPr` values $c(0, 0, 0.5, 0, 0)$ (i.e., the peak prob is 0.5 and the summed prob is 0.5) is less likely to be a changepoint compared to another window values $c(0.1, 0.2, 0.3, 0.2, 0.1)$ (i.e., the peak prob is 0.3 but the summed prob is 0.8).
- `order`: a vector of length *N*; the average harmonic order needed to approximate the seasonal component. As an average over many sampled individual piece-wise harmonic curves, `order` is not necessarily an integer.
- `cp`: [Changepoints] a vector of length $\text{scp.max}=\text{scp.minmax}[2]$; the most possible changepoint locations in the seasonal component. The locations are obtained by first applying a sum-filtering to the `cpOccPr` curve with a filter window size of `sseg.min` and then picking up to a total `ncp` of the highest peaks in the filtered curve. If `ncp<scp.max`, the remaining of the vector is filled with NaNs.
- `cpPr`: [Changepoints PRobability] a vector of length `scp.max`; the probabilities associated with the changepoints `cp`. Filled with NaNs for the remaning elements if `ncp<scp.max`.
- `cpCI`: [Changepoints Credible Interval] a matrix of dimension `scp.max x 2`; the credible intervals for the detected changepoints `cp`.
- `cpAbruptChange`: [Abrupt change at Changepoints] a vector of length `scp.max`; the jumps in the fitted trend curves at the detected changepoints `cp`.
- `Y`: a vector of length *N*; the estimated trend component. It is the Bayesian model averaging of all the individual sampled trend.
- `SD`: [Standard Deviation] a vector of length *N*; the estimated standard deviation of the estimated trend component.
- `CI`: [Standard Deviation] a matrix of dimension *N* x 2; the estimated credible interval of the estimated trend. One vector of the matrix is for the upper envelope and another for the lower envelope.
- `amp`: [AMPlitude] a vector of length *N*; the time-varying amplitude of the estimated seasonality.
- `ampSD`: [Standar Deviation of AMPlitude] a vector of length *N*; , the SD of the amplitude of the seasonality.
- `pos_ncp`:
- `neg_ncp`:

- pos_ncpPr:
- neg_ncpPr:
- pos_cpOccPr:
- neg_cpOccPr:
- pos_cp:
- neg_cp:
- pos_cpPr:
- neg_cpPr:
- pos_cpAbruptChange:
- neg_cpAbruptChange:
- pos_cpCI:
- neg_cpCI: The above variables have the same outputs as those variables without the prefix 'pos' and 'neg', except that we differentiate the change-points with a POSitive jump in the trend from those changepoints with a NEGative jump. For example, pos_ncp refers to the average number of trend changepoints that jump up (i.e., postively) in the trend.

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

See Also

[beast](#), [beast123](#), [minesweeper](#), [tetris](#), [geeLandsat](#)

Examples

```
library(Rbeast)
```

```
#####
# 'ohio' is a data.frame on an irregular Landsat time series of reflectances
# ndvi (e.g., surface greenness) at an Ohio site. It has multiple columns
# of alternative time/date formats, such as year, month, day, doy (date of year),
```

```

# rdate (R's date class), and time (fractional year)

data(ohio)
str(ohio)

#####
# beast.irreg accepts irregular time series and aggregates them into regular ones prior
# to model fitting. For the aggregation, both "time" and "deltat" are needed to specify
# individual times of data points and the regular time interval desired. If there is
# a cyclic componet, freq in the unit of time intervals/datapoints should also be given;
# if not, a possible value is guessed via auto-correlation

#####
# Below, 'time' is given as numeric values, which can be of any arbitray unit. Although
# here 1/12 can be interepreted as 1/12 year or 1 month, beast itself doesn't care about
# the time unit. So, the unit of 1/12 is irrelevant for BEAST. 'freq' is missing and a
# guess of it is used.

o=beast.irreg(ohio$ndvi, time=ohio$time,deltat=1/12)
plot(o)
print(o)

#####
# Explictly provide 'freq'

o=beast.irreg(ohio$ndvi, time=ohio$time,deltat=1/12, freq=12)

#####
# Aggregate the time series at a half-monthly time interval

o=beast.irreg(ohio$ndvi, time=ohio$time,deltat=1/24, freq=24)

#####
# 'time' is given as R's dates. The unit is YEAR. 1/12 refers to 1/12 year or 1 month

o=beast.irreg(ohio$ndvi, time=ohio$rdate,deltat=1/12)

#####
# If the input time has formats (e.g., date strings) other than the above two, use the
# beast123() function instead.

```

Description

A Bayesian model averaging algorithm called BEAST to decompose time series or 1D sequential data into individual components: Abrupt changes, trends, or periodic/seasonal variations. As a multi-purpose tool, it is useful for nonlinear trend analysis, time series decomposition, time series segmentation, changepoint detection, and detection of breakpoints or structural breaks.

Usage

```
beast123( Y,
          metadata=list(),
          prior    =list(),
          mcmc     =list(),
          extra    =list(),
          season   =c('harmonic', 'dummy', 'none'),
          ...)
```

Arguments

- Y** a 1D vector, 2D matrix, or 3D array of numeric data. Missing values are allowed and can be indicated by NA, NaN, or a value customized in the 2nd argument `metadata` (e.g., `metadata$missingValue=-9999`).
- If `Y` is a vector of size $N \times 1$ or $1 \times N$, it is treated as a single time series of length N .
 - If `Y` is a 2D matrix or 3D array of dimension $N1 \times N2$ or $N1 \times N2 \times N3$ (e.g., stacked images of geophysical data), it includes multiple time series of equal length: Which dimension is time has to be given in the 2nd argument using `metadata$whichDimIsTime`. For example, `metadata$whichDimIsTime=1` for a 190×35 input indicates 35 time series of length 190 each; `metadata$whichDimIsTime = 2` for a $100 \times 200 \times 300$ input indicates $30000 = 100 * 300$ time series of length 200 each.
- `Y` can be either regular (i.e., evenly-spaced in time) or irregular/unordered in time.
- If regular, the individual times will be determined from the time of the 1st data point `startTime` and the time span between consecutive points `deltaTime`, which are specified in the 2nd arg through `metadata$startTime` and `metadata$deltaTime`; if not given, `startTime` and `deltaTime` take a default 1.0.
 - If irregular or regular but unordered, the times have to be explicitly given through `metadata$time`. The BEAST model is currently formulated for regular data only, so internally, the `beast` function will aggregate/re-bin irregular data into regular ones; for the aggregation, the `metadata$deltaTime` parameter **MUST** also be also provided to specify the desired bin size or time interval.
- `Y` can have a periodic component or have a trend component only. Use the argument `season` to specify the cases.
- If `season='none'`, `Y` is treated as trend-only; no periodic components are present in the time series.

- If `season='harmonic'`, Y has a periodic/seasonal component. The term 'season' is a misnomer being used here to broad refer to any periodic variations present in Y . The periodicity is not a statistical parameter estimated by `beast` but a known constant given by the user through `metadata$freq`. The periodic component is modeled as a harmonic curve—a combination of sines and cosines.
- If `season='dummy'`, the same as 'harmonic' except that the periodic/seasonal component is modeled as a non-parametric curve.

`metadata`

(optional). If present, `metadata` can be either an `INTEGER` specifying the known period of the cyclic/seasonal component or a `LIST` specifying various parameters to describe the 1st argument Y . If missing, default values will be used, but `metadata` must be explicitly specified if the input Y is a 2D matrix or 3D array. `metadata` is not part of BEAST's Bayesian formulation but just some additional info to interpret Y . Below are possible fields in `metadata`; not all of them are always needed, depending on the types of inputs (e.g., 1D, 2D or 3D; regular or irregular).

- `metadata$whichDimIsTime`: integer (≤ 3). Needed to specify which dimension of Y is time for a matrix or 3D array input. Ignored if the input Y is a vector.
- `metadata$isRegularOrdered`: logical (default to `TRUE` if missing). If `TRUE`, the 1st argument Y is assumed to be regular and if `FALSE`, Y is irregular or regular but unordered in time.
- `metadata$time`: numeric or string values to specify the times for irregular Y . Needed ONLY if `isRegularOrdered = FALSE` (i.e. irregular inputs). Ignored if `isRegularOrdered = TRUE` (i.e., regular data for which `metadata$startTime` and `metadata$deltaTime` are used instead). Three ways of specifying `metadata$time` are supported:
 1. a vector of numerical values to indicate times. The unit of the times is irrelevant to BEAST as long as it remains consistent as the unit used for specifying other variables such as `startTime` and `deltaTime`. The length of the `metadata$time` vector must be the same as Y 's time dimension.
 2. a list of vectors to specify individual dates of the time series. Use `metadata$time$year`, `metadata$time$month`, and `metadata$time$day` to give the dates; or alternatively use `metadata$time$year` and `metadata$time$doy` where each value of the `doy` vector is a number within 1 and 365/366. Each vector must have the same length as the time dimension of Y .
 3. a vector of date strings. Use `metadata$time$dateStr` for the date strings, and `metadata$time$strFmt` to specify the format for parsing `dateStr`. Three formats are currently supported:
 - (a). All the date strings have a fixed pattern in terms of the relative positions of year, month, and day. For example, to extract 2001/12/02 etc from `metadata$time$dateStr = c('P23R34-2001.1202333xd', 'O93X94-2002.1108133fd', 'TP3R34-2009.0122333td')` use `strFmt='P23R34-yyyy.mmdd33'` where `yyyy`, `mm`, and `dd` are the specifiers and other positions are wildcards and can be filled with any other letters different from `yyyy`, `mm` and `dd`.

- (b). All the date strings have a fixed pattern in terms of the relative positions of year and doy. For example, to extract 2001/045(day of year) from 'P23R342001888045', use `strFmt='123123yyyy888doy'` where `yyyy` and `doy` are the specifiers and other positions are wildcards and can be filled with any other letters different from `yyyy`, and `doy`. 'doy' must be three digit in length.
- (c). All the date strings have a fixed pattern in terms of the separation characters between year, month, and day. For example, to extract 2002/12/02 from '2002,12/02', ' 2002 , 12/2', '2002,12 /02', use `strFmt='Y,M/D'` where the whitespaces are ignored. To get 2002/12/02 from '2-12, 2012', use `strFmt='D-M,Y'`.

- `metadata$startTime`: numeric (default to 1.0 if missing). It gives the time of the 1st data point. It can be specified as a scalar (e.g., 2021.23) or a vector of three values in the order of year, month, and day (e.g., `metadata$startTime = c(2021, 1, 24)`). `metadata$startTime` is needed for regular input data but optional for irregular data: If missing, `startTime` will be computed from `metadata$time` for irregular Y.
- `metadata$deltaTime`: numeric. It specifies the time interval between consecutive data points. It is optional for regular data (default to 1.0 if not supplied), but has to be specified for irregular data because `deltaTime` is needed to aggregate/resample the irregular time series into regular ones.
- `metadata$freq`: integer. Needed only for data with a periodic/cyclic component (i.e., `season='harmonic'` or 'dummy') and ignored for trend-only data (i.e., `season='none'`). The "freq" parameter must be an INTEGER specifying the number of samples/values/points per cycle (e.g, a monthly time series with an annual period has a frequency of 12. If `freq` is absent, BEAST first attempts to guess its value via auto-correlation before fitting the model.
- `metadata$missingValue`: numeric; a customized value to indicate bad/missing values in the time series, in addition to those NA or NaN values.
- `metadata$maxMissingRate`: a fractional number within [0, 1] as the maximum percentage of missing values, above which the time series will be skipped and won't be fitted by BEAST.

prior

(optional). a list object consisting of the hyperprior parameters in the Bayesian formulation of the BEAST model. Because they are part of the model, the fitting result may be sensitive to the choices of these hyperparameters. If prior is missing, a set of default values will be used and the exact values used will be printed to the console at the start of the BEAST run. Below are possible parameters:

- `prior$seasonMinOrder`: integer (≥ 1)
- `prior$seasonMaxOrder`: integer (≥ 1); the min and max harmonic orders considered to fit the seasonal component. `seasonMinOrder` and `seasonMaxOrder` are only used if the time series has a seasonal component (i.e., `season='harmonic'`) and ignored for trend-only data or when `season='dummy'`. If `seasonMinOrder=seasonMaxOrder`, BEAST assumes a constant harmonic order used and won't infer the posterior probability of harmonic orders.
- `prior$seasonMinKnotNum`: integer (≥ 0)

- `prior$seasonMaxKnotNum`: integer (≥ 0); the min and max number of seasonal changepoints allowed in segmenting and fitting the seasonal component. `seasonMinKnotNum` and `seasonMaxKnotNum` are only used if the time series has a seasonal component (i.e., `season='harmonic'` or `season='dummy'`) and ignored for trend-only data. If `seasonMinOrder=seasonMaxOrder`, BEAST assumes a constant number of changepoints and won't infer the posterior probability of the number of changepoints, but it will still estimate the occurrence probability of the changepoints over time (i.e., the most likely times at which these changepoints occur). If `seasonMinOrder=seasonMaxOrder=0`, no changepoints are allowed in the seasonal component; then a global harmonic model is used to fit the seasonal component.
- `prior$seasonMinSepDist`: integer (> 0). the min separation time between two neighboring season changepoints. That is, when fitting a piecewise harmonic seasonal model, no two changepoints are allowed to occur within a time window of `seasonMinSepDist`. `seasonMinSepDist` must be a unitless integer—the number of time intervals/data points so that the time window in the original unit is `seasonMinSepDist*metadata$deltaTime`.
- `prior$trendMinOrder`: integer (≥ 0)
- `prior$trendMaxOrder`: integer (≥ 0); the min and max orders of the polynomials considered to fit the trend component. The zero-th order corresponds to a constant term/ a flat line and the 1st order is a line. If `trendMinOrder=trendMaxOrder`, BEAST assumes a constant polynomial order used and won't infer the posterior probability of polynomial orders.
- `prior$trendMinKnotNum`:
- `prior$trendMaxKnotNum`: integer (≥ 0); the min and max number of trend changepoints allowed in segmenting and fitting the trend component. If `trendMinOrder=trendMaxOrder`, BEAST assumes a constant number of changepoints in the fitted trend and won't infer the posterior probability of the number of trend changepoints, but it will still estimate the occurrence probability of the changepoints over time (i.e., the most likely times at which these changepoints occur). If `trendMinOrder=trendMaxOrder=0`, no changepoints are allowed in the trend component; then a global polynomial model is used to fit the trend.
- `prior$trendMinSepDist`: integer (> 0). the min separation time between two neighboring trend changepoints.
- `prior$precValue`: numeric (> 0); the default value is 10.
- `prior$precPriorType`: characters. It takes one of 'constant', 'uniform' (the default), 'componentwise', and 'orderwise'. Below are the differences between them.
 1. `precPriorType='constant'`: the precision parameter used to parameterize the model coefficients is fixed to a constant specified by `prior$precValue`. In other words, `prior$precValue` is a user-defined hyperparameter and the fitting result may be sensitive to the chosen values of `prior$precValue`.
 2. `precPriorType='uniform'`: the precision parameter used to parameterize the model coefficients is a random variable; its initial value is specified by `prior$precValue`. In other words, `precValue` will be inferred by the MCMC, so the fitting result is insensitive to the choice in `prior$precValue`.

3. `precPriorType='componentwise'`: multiple precision parameters are used to parameterize the model coefficients for individual components (e.g., one for season and another for trend); their initial values is specified by `prior$precValue`. In other words, `precValue` will be inferred by the MCMC, so the fitting result is insensitive to the choice in `prior$precValue`.
4. `precPriorType='orderwise'`: multiple precision parameters are used to parameterize the model coefficients not just for individual components but also for individual orders of each component; their initial values is specified by `prior$precValue`. In other words, `precValue` will be inferred by the MCMC, so the fitting result is insensitive to the choice in `prior$precValue`.

mcmc

(optional). a list object consisting of parameters to configure the MCMC inference. These parameter are not part of the Bayesian formulation of the BEAST model but are the settings for the reversible-jump MCMC to generate MCMC chains. Due to the MCMC nature, the longer the simulation chain is, the better the fitting result. Below are possible parameters:

- `mcmc$seed`: integer (≥ 0); the seed for the random number generator. If `mcmc$seed=0`, an arbitrary seed will be picked up and the fitting result will var across runs. If fixed to the same on-zero integer, the results can be re-produced for different runs. Note that the results may still vary if run on different computers with the same seed because the random generator library depends on CPU's instruction sets.
- `mcmc$samples`: integer (> 0); the number of samples collected per MCMC chain.
- `mcmc$chainNumber`: integer (> 0); the number of parallel MCMC chains.
- `mcmc$thinningFactor`: integer (> 0); a factor to thin chains (e.g., if `thinningFactor=5`, samples will be taken every 3 iterations).
- `mcmc$burnin`: integer (> 0); the number of burn-in samples discarded at the start of each chain.
- `mcmc$maxMoveStepSize`: integer (> 0). The RJMCMC sampler employs a move proposal when traverseing the model space or proposing new positions of changepoints. `'maxMoveStepSize'` is used in the move proposal to specify the max window allowed in jumping from the current changepoint.
- `mcmc$seasonResamplingOrderProb`: a fractional number less than 1.0; the probability of selecting a re-sampling proposal (e.g., resample seasonal harmonic order).
- `mcmc$trendResamplingOrderProb`: a fractional number less than 1.0; the probability of selecting a re-sampling proposal (e.g., resample trend polynomial order)
- `mcmc$credIntervalAlphaLevel`: a fractional number less than 1.0 (default to 0.95); the level of confidence used to compute credible intervals.

extra

(optional). a list object consisting of flags to control the outputs from the BEAST runs or configure other program setting. Below are possible parameters:

- `extra$dumpInputData`: logical (default to FALSE). If TRUE, the input time series will be copied into the output. When the input Y is irregular (i.e.,

metadata\$isRegularOrdered=FALSE), the dumped copies will be the aggregated regular time series.

- extra\$whichOutputDimIsTime: integer (≤ 3). If the input Y is a 2D or 3D array (i.e., multiple time series such as stacked images), the whichOutputDimIsTime specifies which dimension is the time in the output variables. whichOutputDimIsTime defaults to 3 for 3D inputs and is ignored if the input is a vector (i.e., a single time series).
- extra\$computeCredible: logical (default to TRUE). Credible intervals will be computed and outputted only if set to TRUE.
- extra\$fastCIComputation: logical (default to TRUE). If TRUE, a fast method is used to compute credible intervals (CI). Computation of CI is one of the most computational parts and fastCIComputation should be set to TRUE unless more accurate CI estimation is desired.
- extra\$computeSeasonOrder: logical (default to TRUE). If TRUE, a posterior estimate of the seasonal harmonic order will be outputted; this flag is only valid if the time series has a seasonal component (i.e., season='harmonic' and prior\$seasonMinOrder is not equal to prior\$seasonMaxOrder).
- extra\$computeTrendOrder: logical (default to TRUE). If TRUE, a posterior estimate of the trend polynomial order will be outputted; this flag is only valid when prior\$trendMinOrder is not equal to prior\$trendMaxOrder).
- extra\$computeTrendOrder: logical (default to TRUE). If TRUE, a posterior estimate of the trend polynomial order will be outputted; this flag is only valid when prior\$trendMinOrder is not equal to prior\$trendMaxOrder).
- extra\$computeSeasonChngpt: logical (default to TRUE). If TRUE, compute the most likely times/positions where changepoints occur in the seasonal component. This flag is not valid if there is a seasonal component in the time series (i.e., season='harmonic' or season='dummy' and prior\$seasonMaxKnotNum is non-zero).
- extra\$computeTrendChngpt: logical (default to TRUE). If TRUE, compute the most likely times/positions where changepoints occur in the trend component.
- extra\$computeSeasonAmp: logical (default to FALSE). If TRUE, compute and output the time-varying amplitude of the seasonality.
- extra\$computeTrendSlope: logical (default to FALSE). If TRUE, compute and output the time-varying slope of the estimated trend.
- extra\$tallyPosNegSeasonJump: logical (default to FALSE). If TRUE, compute and differentiate seasonal changepoints in terms of the direction of the jumps in the estimated seasonal signal. Those changepoints with a positive jump will be outputted separately from those with a negative jump. A series of output variables (some for positive-jump changepoints, and others for negative-jump changepoints) will be dumped.
- extra\$tallyPosNegTrendJump: logical (default to FALSE). If TRUE, compute and differentiate trend changepoints in terms of the direction of the jumps in the estimated trend. Those changepoints with a positive jump will be outputted separately from those with a negative jump. A series of output variables (some for positive-jump changepoints, and others for negative-jump changepoints) will be dumped.

- `extra$tallyIncDecTrendJump`: logical (default to FALSE). If TRUE, compute and differentiate trend changepoints in terms of the direction of the jumps in the estimated slope of the trend signal. Those changepoints with a increase in the slope will be outputted separately from those with a decrease in the slope. A series of output variables (some for increase-jump changepoints, and others for decrease-jump changepoints will be dumped).
 - `extra$printProgressBar`: logical (default to FALSE). If TRUE, a progress bar will be displayed to show the status of the running. When running on multiple time series (e.g. stacked image time series), the progress bar will also report an estimate of the remaining time for completion.
 - `extra$consoleWidth`: integer (default to 0); the length of chars in each status line when setting `printProgressBar=TRUE`. If 0, the current width of the console will be used.
 - `extra$printOptions`: logical (default to FALSE). If TRUE, the values used in the arguments `metadata`, `prior`, `mcmc`, and `extra` will be printed to the console at the start of the run.
 - `extra$numThreadsPerCPU`: integer (default to 2); the number of threads to be scheduled for each CPU core.
 - `extra$numParThreads`: integer (default to 0). When handling many time series, BEAST can use multiple concurrent threads. `extra$numParThreads` specifies how many concurrent threads will be used in total. If `numParThreads=0`, the actual number of threads will be `numThreadsPerCPU * cpuCoreNumber`; that is, each CPU core will generate a number 'numThreadsPerCPU' of threads. On Windows 64, BEAST is group-aware and will affine the threads to all the NUMA node. But currently, up to 256 CPU cores are supported.
- `season` characters (default to 'harmonic'); specify if `y` has a periodic component or not. Three strings are possible.
- 'none': `y` is trend-only; no periodic components are present in the time series. The args for the seasonal component (i.e., `sorder.minmax`, `scp.minmax` and `sseg.max`) will be irrelevant and ignored.
 - 'harmonic': `y` has a periodic/seasonal component. The term `season` is a misnomer, being used here to broadly refer to any periodic variations present in `y`. The periodicity is NOT a model parameter estimated by BEAST but a known constant given by the user through `freq`. By default, the periodic component is modeled as a harmonic curve—a combination of sines and cosines.
 - If 'dummy', the same as 'harmonic' except that the periodic/seasonal component is modeled as a non-parametric curve. The harmonic order arg `sorder.minmax` is irrelevant and is ignored.
- ... additional parameters, not used currently but reserved for future extension

Value

The output is an object of class "beast". It is a list, consisting of the following variables. Exact sizes of the variables depend on the types of the input `Y` as well as the specified output time dimension `extra$whichOutputDimIsTime`. In the explanations below, we assume the input `Y` is a single time series of length `N`; the dimensions for 2D or 2D inputs may be interpreted accordingly:

time	a vector of size 1xN: the times at the N sampled locations. By default, it is simply set to 1:N
data	a vector, matrix, or 3D array; this is a copy of the input Y if <code>extra\$dumpInputData = TRUE</code> . If <code>extra\$dumpInputData=FALSE</code> , it is set to NULL. If the original input Y is irregular, the copy here is the regular version aggregated from the original at the time interval specified by <code>metadata\$deltaTime</code> .
marg_lik	numeric; the average of the model marginal likelihood; the larger <code>marg_lik</code> , the better the fitting for a given time series.
R2	numeric; the R-square of the model fitting.
RMSE	numeric; the RMSE of the model fitting.
sig2	numeric; the estimated variance of the model error.
trend	a list object numeric consisting of various outputs related to the estimated trend component: <ul style="list-style-type: none"> • <code>ncp</code>: [Number of ChangePoints]. a numeric scalar; the mean number of trend changepoints. • <code>ncpPr</code>: [Probability of the Number of ChangePoints]. A vector of length <code>(prior\$trendMaxKnotNum+1)</code>. It gives a probability distribution of having a certain number of trend changepoints over the range of <code>[0,prior\$trendMaxKnotNum]</code>; for example, <code>ncpPr[1]</code> is the probability of having no trend changepoint; <code>ncpPr[i]</code> is the probability of having <code>(i-1)</code> changepoints: Note that it is <code>ncpPr[i]</code> not <code>ncpPr[i-1]</code> because <code>ncpPr[1]</code> is used for having zero changepoint. • <code>cpOccPr</code>: [ChangePoint OCCurrence PRObability]. a vector of length N; it gives a probability distribution of having a changepoint in the trend at each point of time. Plotting <code>cpOccPr</code> will depict a continuous curve of probability-of-being-changepoint. Of particular note, in the curve, a higher peak indicates a higher chance of being a changepoint only at that particular SINGLE point in time and does not necessarily mean a higher chance of observing a changepoint AROUND that time. For example, a window of <code>cpOccPr</code> values <code>c(0, 0, 0.5, 0, 0)</code> (i.e., the peak prob is 0.5 and the summed prob is 0.5) is less likely to be a changepoint compared to another window <code>c(0.1, 0.2, 0.21, 0.2, 0.1)</code> (i.e., the peak prob is 0.21 but the summed prob is 0.71). • <code>order</code>: a vector of length N; the average polynomial order needed to approximate the fitted trend. As an average over many sampled individual piece-wise polynomial trends, <code>order</code> is not necessarily an integer. • <code>cp</code>: [Changepoints] a vector of length <code>metadata\$trendMaxKnotNum</code>; the most possible changepoint locations in the trend component. The locations are obtained by first applying a sum-filtering to the <code>cpOccPr</code> curve with a filter window size of <code>prior\$trendMinSeptDist</code> and then picking up to a total <code>ncp</code> of the highest peaks in the filtered curve. If <code>ncp < trendMaxKnotNum</code>, the remaining of the vector is filled with NaNs. • <code>cpPr</code>: [Changepoints PRObability] a vector of length <code>metadata\$trendMaxKnotNum</code>; the probabilities associated with the changepoints <code>cp</code>. Filled with NaNs for the remaining elements if <code>ncp < trendMaxKnotNum</code>.

- cpCI: [Changepoints Credible Interval] a matrix of dimension `metadata$trendMaxKnotNum` x 2; the credible intervals for the detected changepoints cp.
- cpAbruptChange: [Abrupt change at Changepoints] a vector of length `metadata$trendMaxKnotNum`; the jumps in the fitted trend curves at the detected changepoints cp.
- Y: a vector of length N; the estimated trend component. It is the Bayesian model averaging of all the individual sampled trend.
- SD: [Standard Deviation] a vector of length N; the estimated standard deviation of the estimated trend component.
- CI: [Standard Deviation] a matrix of dimension N x 2; the estimated credible interval of the estimated trend. One vector of the matrix is for the upper envelope and another for the lower envelope.
- slp: [Slope] a vector of length N; the time-varying slope of the fitted trend component .
- slpSD: [Standar Deviation of Slope] a vector of length N; the SD of the slope for the trend component.
- slpSignPr: [PRobability of slope having a postive sign] a vector of length N; the probability of the slope being positive (i.e., increasing trend) for the trend component. For example, if `slpSignPr=0.80` at a given point in time, it means that 80
- pos_ncp:
- neg_ncp:
- pos_ncpPr:
- neg_ncpPr:
- pos_cpOccPr:
- neg_cpOccPr:
- pos_cp:
- neg_cp:
- pos_cpPr:
- neg_cpPr:
- pos_cpAbruptChange:
- neg_cpAbruptChange:
- pos_cpCI:
- neg_cpCI: The above variables have the same outputs as those variables without the prefix 'pos' and 'neg', except that we differentiate the changepoints with a POStive jump in the trend from those changepoints with a NEGative jump. For example, `pos_ncp` refers to the average number of trend changepoints that jump up (i.e., postively) in the trend.
- inc_ncp:
- dec_ncp:
- inc_ncpPr:
- dec_ncpPr:
- inc_cpOccPr:
- dec_cpOccPr:
- inc_cp:

- `dec_cp`:
- `inc_cpPr`:
- `dec_cpPr`:
- `inc_cpAbruptChange`:
- `dec_cpAbruptChange`:
- `inc_cpCI`:
- `dec_cpCI`: The above variables have the same outputs as those variables without the prefix 'inc' and 'dec', except that we differentiate the change-points at which the trend slope increases from those changepoints at which the trend slope decreases. For example, if the trend slopes before and after a `chnp` is 0.4 and 2.5, then the changepoint is counted toward `inc_ncp`.

`season`

a list object numeric consisting of various outputs related to the estimated seasonal/periodic component:

- `ncp`: [Number of ChangePoints]. a numeric scalar; the mean number of seasonal changepoints.
- `ncpPr`: [Probability of the Number of ChangePoints]. A vector of length $(\text{prior}\$seasonMaxKnotNum+1)$. It gives a probability distribution of having a certain number of seasonal changepoints over the range of $[0, \text{prior}\$seasonMaxKnotNum]$; for example, `ncpPr[1]` is the probability of having no seasonal change-point; `ncpPr[i]` is the probability of having (i-1) changepoints: Note that the index is i rather than (i-1) because `ncpPr[1]` is used for having zero changepoint.
- `cpOccPr`: [ChangePoint OCCurrence PRObability]. a vector of length N; it gives a probability distribution of having a changepoint in the seasonal component at each point of time. Plotting `cpOccPr` will depict a continuous curve of probability-of-being-changepoint over the time. Of particular note, in the curve, a higher value at a peak indicates a higher chance of being a changepoint only at that particular SINGLE point in time, and does not necessarily mean a higher chance of observing a changepoint AROUND that time. For example, a window of `cpOccPr` values $c(0, 0, 0.5, 0, 0)$ (i.e., the peak prob is 0.5 and the summed prob is 0.5) is less likely to be a change-point compared to another window values $c(0.1, 0.2, 0.3, 0.2, 0.1)$ (i.e., the peak prob is 0.3 but the summed prob is 0.8).
- `order`: a vector of length N; the average harmonic order needed to approximate the seasonal component. As an average over many sampled individual piece-wise harmonic curves, `order` is not necessarily an integer.
- `cp`: [Changepoints] a vector of length `metadata$seasonMaxKnotNum`; the most possible changepoint locations in the seasonal component. The locations are obtained by first applying a sum-filtering to the `cpOccPr` curve with a filter window size of `prior$trendMinSeptDist` and then picking up to a total `ncp` of the highest peaks in the filtered curve. If `ncp < seasonMaxKnotNum`, the remaining of the vector is filled with NaNs.
- `cpPr`: [Changepoints PRObability] a vector of length `metadata$seasonMaxKnotNum`; the probabilities associated with the changepoints `cp`. Filled with NaNs for the remaining elements if `ncp < seasonMaxKnotNum`.

- cpCI: [Changepoints Credible Interval] a matrix of dimension $\text{metadata}\$\text{seasonMaxKnotNum} \times 2$; the credible intervals for the detected changepoints cp.
- cpAbruptChange: [Abrupt change at Changepoints] a vector of length $\text{metadata}\$\text{seasonMaxKnotNum}$; the jumps in the fitted trend curves at the detected changepoints cp.
- Y: a vector of length N; the estimated trend component. It is the Bayesian model averaging of all the individual sampled trend.
- SD: [Standard Deviation] a vector of length N; the estimated standard deviation of the estimated trend component.
- CI: [Standard Deviation] a matrix of dimension $N \times 2$; the estimated credible interval of the estimated trend. One vector of the matrix is for the upper envelope and another for the lower envelope.
- amp: [AMplitude] a vector of length N; the time-varying amplitude of the estimated seasonality.
- ampSD: [Standard Deviation of AMplitude] a vector of length N; , the SD of the amplitude of the seasonality.
- pos_ncp:
- neg_ncp:
- pos_ncpPr:
- neg_ncpPr:
- pos_cpOccPr:
- neg_cpOccPr:
- pos_cp:
- neg_cp:
- pos_cpPr:
- neg_cpPr:
- pos_cpAbruptChange:
- neg_cpAbruptChange:
- pos_cpCI:
- neg_cpCI: The above variables have the same outputs as those variables without the prefix 'pos' and 'neg', except that we differentiate the changepoints with a POSitive jump in the trend from those changepoints with a NEGative jump. For example, pos_ncp refers to the average number of trend changepoints that jump up (i.e., positively) in the trend.

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).

- Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. ISPRS Journal of Photogrammetry and Remote Sensing, 176, pp.250-261(a beast application paper).

See Also

[beast](#), [beast.irreg](#), [minesweeper](#), [tetris](#), [geLandsat](#)

Examples

```
library(Rbeast)
#####
# beast123 is an all-inclusive function that duplicates the functionalities of beast
# and beast.irreg. It can handle a single, multiple, or 3D of stacked time series, being
# either regular or irregular. It allows for customization through four LIST option
# arguments:
# metadata -- additional info about the input Y
# prior    -- prior parameters for the beast model
# mcmc     -- MCMC simulation setting
# extra    -- misc parameters turning on/off outputs and setting up parallel computations
#####
# Yellowstone is a half-monthly time series of 774 NDVI measurmments at a Yellowstone
# site starting from July 1-15,1981(i.e., start=c(1981,7,7)). It has 24 data points per
# year (freq=24).

data(Yellowstone)
plot(Yellowstone)

#####
# The four option args are missing, so defalut values will be used, with some warning
# messages given to alert this. By default, the input Y is assumed to be regular with
# a seasonal component. The default values used will be printed out and they can serve
# as a template to customize the parameters.

o = beast123(Yellowstone)
plot(o)

#####
# Nile is an annual river flow time series (i.e., no periodic variation). So, season
# is set to 'none' for trend-only analysis. Default values are used for other
# missing options. Unlike the beast function, beast123 does NOT use the time attributes
# of a 'ts' object. For example, Nile is treated as a pure data number; its (start=1871,
# end=1970, freq=1) attributes are ignored. The default times 1:length(Nile) are used
# instead. The true time info need to be specified by the metadata parameter, as shown
# in the next example.

o = beast123(Nile,season='none')
plot(o)
```

```
#####
# Specify metadata, prior, mcmc, and extra explicitly. Only 'prior' is the true model
# parameters of BEAST; the other three are just options to configure the input/output or
# the computation process.

# metadata is NOT part of BEAST itself, but some extra info to describe the input
# time series Y. Below, the input Y is the 'Yellowstone' ts.
metadata = list()
metadata$isRegularOrdered = TRUE      # Regular input
metadata$whichDimIsTime = 1          # Which dim of the input refer to time for
                                     # 2D/3D inputs? Ignored for a single time
                                     # series input.
metadata$startTime = c(1981,7,7)    # Or startTime=1981.5137
                                     #   startTime=as.Date('1981-7-7')
metadata$deltaTime = 1/24           # Half-monthly regular ts: 0.5/12=1/24
metadata$period = 1.0               # The period is 1 year:
                                     # freq x deltaTime = period
                                     # 24 x 1/24 = 1.0
metadata$omissionValue = NaN        # By default, NaNs are ignored
metadata$maxMissingRateAllowed = 0.7500 # If missingness is higher than .75, the ts
                                     # is skipped and not fitted
metadata$deseasonalize = FALSE      # Do not remove the global seasonal pattern
                                     # before fitting the beast model
metadata$detrend = FALSE            # Do not remove the global trend before
                                     # the fitting

# prior is the ONLY true parameters of the beast model,used to specify the priors
# in the Bayesian formulation
prior = list()
prior$seasonMinOrder = 1            #min harmonic order allowed to fit seasonal cmpnt
prior$seasonMaxOrder = 5            #max harmonic order allowed to fit seasonal cmpnt
prior$seasonMinKnotNum = 0          #min number of changepts in seasonal cmpnt
prior$seasonMaxKnotNum = 3          #max number of changepts in seasonal cmpnt
prior$seasonMinSepDist = 10         #min inter-chngpts seperation for seasonal cmpnt
prior$trendMinOrder = 0             #min polynomial order allowed to fit trend cmpnt
prior$trendMaxOrder = 1            #max polynomial order allowed to fit trend cmpnt
prior$trendMinKnotNum = 0          #min number of changepts in trend cmpnt
prior$trendMaxKnotNum = 15         #max number of changepts in trend cmpnt
prior$trendMinSepDist = 5          #min inter-chngpts seperation for trend cmpnt
prior$precValue = 10.0             #Initial value of the precision parameter (no
                                     # need to change it unless for precPrioType='const')
prior$precPriorType = 'uniform'    # Possible values: const, uniform, and componentwise

# mcmc is NOT part of the beast model itself, but some parameters to configure the
# MCMC inference.
mcmc = list()

mcmc$seed = 9543434                # an arbitray seed for random number generator
mcmc$samples = 3000                # samples collected per chain
mcmc$thinningFactor = 3            # take every 3rd sample and discard others
mcmc$burnin = 150                  # discard the initial 150 samples per chain
```

```

mcmc$chainNumber          = 3      # number of chains
mcmc$maxMoveStepSize      = 4      # max random jump step when proposing new chngpts
mcmc$trendResamplingOrderProb = 0.100 # prob of choosing to resample polynomial order
mcmc$seasonResamplingOrderProb = 0.100 # prob of choosing to resample harmonic order
mcmc$credIntervalAlphaLevel = 0.950 # the significance level for credible interval

# extra is NOT part of the beast model itself, but some parameters to configure the
# output and computation process
extra = list()
extra$dumpInputData      = FALSE #If true, a copy of input time series is outputted
extra$whichOutputDimIsTime = 1    #For 2D or 3D inputs, which dim of the output refers to
                                # time? Ignored if the input is a single time series
extra$computeCredible    = FALSE #If true, compute CI: computing CI is time-intensive.
extra$fastCIComputation  = TRUE   #If true, a faster way is used to get CI, but it is
                                # still time-intensive. That is why the function beast()
                                # is slow because it always compute CI.
extra$computeSeasonOrder = FALSE #If true, dump the estimated harmonic order over time
extra$computeTrendOrder  = FALSE #If true, dump the estimated polynomial order over time
extra$computeSeasonChngpt = TRUE  #If true, get the most likely locations of s chngpts
extra$computeTrendChngpt  = TRUE  #If true, get the most likely locations of t chngpts
extra$computeSeasonAmp    = FALSE #If true, get time-varying amplitude of seasonality
extra$computeTrendSlope   = FALSE #If true, get time-varying slope of trend
extra$tallyPosNegSeasonJump = FALSE #If true, get those changpts with +/- jumps in season
extra$tallyPosNegTrendJump = FALSE #If true, get those changpts with +/- jumps in trend
extra$tallyIncDecTrendJump = FALSE #If true, get those changpts with increasing/
                                # decreasing trend slopes

extra$printProgressBar    = TRUE
extra$printOptions        = TRUE
extra$consoleWidth        = 0    # If 0, the console width is from the current console
extra$numThreadsPerCPU    = 2    # 'numThreadsPerCPU' and 'numParThreads' are used to
extra$numParThreads       = 0    # configure multithreading runs; they're used only if
                                # Y has multiple time series (e.g.,stacked images)

o = beast123(Yellowstone,metadata,prior,mcmc,extra, season='harmonic')
plot(o)

#####
# Handle irregular time series: ohio is a data frame of a Landsat NDVI series observed
# at unevely-spaced times

data(ohio)
str(ohio)

metadata          = list()
metadata$isRegularOrdered = FALSE    # The input data is irregular
metadata$time      = ohio$time # Must supply individual times for irregular inputs
metadata$deltaTime = 1/12    # Must supply the desired time interval for aggregation
metadata$period    = 1.0

o=beast123(ohio$ndvi, metadata)      # Default values used for those missing parameters

```

```
#####
class(ohio$rdate)                # Another accepted time format for beast123

metadata = list()
metadata$isRegularOrdered = FALSE # The input data is irregular
metadata$deltaTime      = 1/12    # Must supply the desired time interval for aggregation
metadata$time           = ohio$rdate # Must supply individual times for irregular inputs

o=beast123(ohio$ndvi, metadata)    # Default values used for those missing parameters

#####
ohio$Y                          # Another accepted time format for beast123
ohio$M
ohio$M

metadata = list()
metadata$isRegularOrdered = FALSE # The input data is irregular
metadata$deltaTime        = 1/12   # Must supply the desired time interval for aggregation
metadata$time$year        = ohio$Y
metadata$time$month       = ohio$M
metadata$time$day         = ohio$D
o=beast123(ohio$ndvi, metadata)    # Default values used for those missing parameters

#####
ohio$Y                          # Another accepted time format for beast123
ohio$doy

metadata = list()
metadata$isRegularOrdered = FALSE # Irregular input
metadata$deltaTime        = 1/12   # Must supply the desired time interval for aggregation
metadata$time$year        = ohio$Y
metadata$time$doy         = ohio$doy
o=beast123(ohio$ndvi, metadata)    # Default values used for those missing parameters

#####
ohio$time                        # Another accepted time format for beast123

metadata = list()
metadata$isRegularOrdered = FALSE # Irregular input
metadata$deltaTime        = 1/12   # Must supply the desired time interval for aggregation
metadata$time             = ohio$time # Fractional year

o=beast123(ohio$ndvi, metadata)    # Default values used for those missing parameters

#####
ohio$datestr1                    # Another accepted time format for beast123

metadata = list()
metadata$isRegularOrdered = FALSE # Irregular input
metadata$deltaTime        = 1/12   # Must supply the time interval for aggregation
metadata$time$datestr     = ohio$datestr1
metadata$time$strfmt      = '???y?mm?dd'
```

```

o=beast123(ohio$ndvi, metadata)          # Default values used for those missing parameters

#####
ohio$datestr2                          # Another accepted time format for beast123
metadata = list()
metadata$isRegularOrdered = FALSE       # Irregular input
metadata$deltaTime = 1/12               # Must supply a desired time interval for aggregation
metadata$time$datestr = ohio$datestr2
metadata$time$strfmt = '???y?y?y?doy???'

o=beast123(ohio$ndvi, metadata)          # Default values used for those missing parameters

#####
ohio$datestr3                          # Another accepted time format for beast123
metadata = list()
metadata$isRegularOrdered = FALSE       # Irregular input
metadata$deltaTime = 1/12               # Must supply the desired time interval for aggregation
metadata$time$datestr = ohio$datestr3
metadata$time$strfmt = 'Y,,M/D'

o=beast123(ohio$ndvi, metadata)          # Default values used for those missing parameters

#####
# Handle multiple time series: 'simdata' is a 2D matrix of dim 300x3; it consists of 3
# time series of length 300 each. As a toy example, the 3 time series are the same.

data(simdata)                          #
dim(simdata)                            # dim of simdata: 300 x 3 (time x num_of_ts)

metadata = list()
metadata$whichDimIsTime = 1              # Which dim of the input refer to time for 2D inputs?
                                          # 300 is the ts length, so dim is set to '1' here.
metadata$period = 24                    # By default, we assume startTime=1 and deltaTime=1

extra=list()
extra$whichOutputDimIsTime = 2          # Which dim of the output arrays refers to time?
o=beast123(simdata, metadata, extra=extra) # Default values used for those missing parameters

##### Another run by transposing simdata

simdata1=t(simdata)                     # dim of simdata1: 3 x 300 (num of ts x time )

metadata = list()
metadata$whichDimIsTime = 2              # Which dim of the input refer to time for 2D inputs?
                                          # 300 is the ts length, so dim is set to '2' here.
metadata$period = 24                    # By default, we assume startTime=1 and deltaTime=1
o=beast123(simdata1, metadata)           # Default values used for those missing parameters

#####

```

```

# Handle 3D stacked images of irregular and unordered time-series: imagestack is a 3D
# array of size 12x9x1066, each pixel being a time series of length 1066

data(imagestack)
dim(imagestack$ndvi)          # Dim: 12 x 9 X 1066 (row x col x time)
imagestack$datestr           # A character vector of 1066 date strings

metadata                      = list()
metadata$isRegularOrdered    = FALSE # 'imagestack$ndvi' is an IRREGULAR input
metadata$whichDimIsTime      = 3     # Which dim of the input refer to time for 3D inputs?
                                # 1066 is the ts length, so dim is set to '3' here.
metadata$time$datestr        = imagestack$datestr
metadata$time$strfmt          = 'LT05_018032_20080311.yyyy-mm-dd'
metadata$deltaTime           = 1/12 # Aggregate the irregular ts at a monthly interval:1/12 Yr
metadata$period               = 1.0  # The period is 1 year: deltaTime*freq=1/12*12=1.0

extra = list()
extra$dumpInputData          = TRUE  # Get a copy of aggregated input ts
extra$numThreadsPerCPU      = 2     # Each cpu core will be assigned 2 threads
extra$numParThreads          = 0     # If 0, total_num_threads=numThreadsPerCPU*num_of_cpu_core
                                # if >0, used to specify the total number of threads

# Default values for missing parameters
o=beast123(imagestack$ndvi, metadata=metadata,extra=extra)

print(o,c(5,3))             # print the result for the pixel at Row 5 and Col 3
plot(o,c(5,3))              # plot the result for the pixel at Row 5 and Col 3
image(o$trend$ncp)          # number of trend changepoints over space

#####
# Handle 3D stacked images of irregular time-series : 'ndvi.zip' is a zip file of
# 437 NDIV tiff image files, each having a dim of 12 x 9.

zipfile = system.file("extdata/ndvi.zip", package="Rbeast") # the full path of ndvi.zip
tmpfld = file.path(tempdir(), "RbeastTmpFld" ) # get a temp folder
dir.create(tmpfld) # create the tmp folder if non-existing
unzip(zipfile, exdir= normalizePath(tmpfld)) # unzip the images to the tmp folder

#library(raster) # uncomment this line when trying this example
filelist = list.files(path=tmpfld, pattern='*.tif', full.names=TRUE) # the tiff filenames
ndvi3d = stack(filelist) # create a rasterstack object
inMemory(ndvi3d) # image data not read yet
ndvi3d = readAll(ndvi3d) # To use beast, make sure all the data is read into memory
# 'raster' is slow with the reading,even for our small images
ndvi3d = brick(ndvi3d) # Convert the stack to a RasterBrick object
inMemory(ndvi3d) # Now, all the image data is in memory
dims = dim(ndvi3d) # WARNING: the 'raster' package reports a dim of 12 x 9 x 437
#corresponding to row x col x band/time. But 'raster' seems to
# use a row-major storage mode. So internally, the RasterBrick image
# data is 9 x 12 x 437 in term of R's column-major storage mode:

```

```

# beast also use the col-major mode.
Y      = values(ndvi3d)
dim(Y) = c(9, 12, 437) # Assign the correct column-major dim
datestr = names(ndvi3d) # the time strings

metadata = list()
metadata$isRegularOrdered = FALSE # IRREGULAR input
metadata$whichDimIsTime = 3 # 437 is the ts length, so dim is set to '3' here.
metadata$time$datestr = datestr
metadata$time$strfmt = 'LT05_018032_20080311.yyyy-mm-dd'
metadata$deltaTime = 1/12 # Aggregate the irr ts at a monthly interval:1/12 year
metadata$period = 1.0 # The period is 1 year: deltaTime*freq=1/12*12=1.0

extra = list()
extra$numThreadsPerCPU = 3 # Each cpu core will spawn 3 threads
extra$numParThreads = 30 # A total of 30 threads used, each processing individual
# time series. For a computer with 100 cores, only the
# first 30/3=10 cores will be used; if the computer has 5
# cores only, then each core will spawn 30/5=6 threads
# instead of numThreadsPerCPU=3.

o=beast123(Y, metadata=metadata,extra=extra) # Default values used for missing parameters

# WARNING: bcz the input ts is from a RasterBrick object that has a row-major storage mode,
# the beast output will be of size 9 x 12 for each band r time. That is, the row and col are
# reversed. This is a glitch only with the raster package. The beast output dim is consistent
# with the dim of the input Y.

image(o$trend$ncp) # number of changepoints over space
grid::grid.raster(o$trend$ncpPr[,1:3]) # an pseduo-rgb image composed from the prob of having
# 0, 1, and 2 trend changepoints.

plot(o,c(4,5))

unlink(tmpfld, recursive = TRUE) # Delete the tmp folder and the extract tiff images

```

CNAchrom11

*DNA copy number alteration data in array-based CGH data for
Chromosome 11*

Description

CNAchrom11 is a vector of the log₂ intensity ratios for cell line GM03576 for Chromosome 11, obtained from Snijders et al. (2001).

Usage

```
data(CNAchrom11)
```

Source

Snijders et al. (2001), Assembly of microarrays for genome-wide measurement of DNA copy number, *Nature Genetics*, 29, 263-264 (<http://www.nature.com/ng/journal/v29/n3/full/ng754.html>).

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

Examples

```
library(Rbeast)
data(CNAchrom11)
```

```
geeLandsat
```

```
Landsat reflectance and NDVI time series from Google Earth Engine
```

Description

Get Landsat reflectance and NDVI time series from Google Earth Engine given longitude and latitude

Usage

```
geeLandsat(lon=NA, lat=NA, radius=100, stat='mean', timeout=700)
```

Arguments

lon	numeric within [-180,180]
lat	numeric within [-90, 90]
radius	a positive number (<=500 meters); the radius of a buffer around the given latitude and longitude for aggregation. If radius=0, the single pixel at the lat and lon will be retrieved

stat	character; if radius>0, used to specify the spatial aggregation method for pixels in the buffer. Possible values are 'mean', 'min', 'max', or 'median'.
timeout	integer; the seconds elapsed to wait for connection timeout. See the note for an explanation.

Value

a data.frame object consisting of dates, sensor type, reflectances, and NDVI for the requested location. It contains only valid and clear-sky values as obtained by referring to the standard clouds flags.

Note

As a poor man's scheme to interact with Google Earth Engine, geeLandsat should be used only for occasional retrieval of Landsat time series at a few sites, NOT for batch downloading for thousands of sites in a R loop. This procedure is provided to get example time series for testing BEAST. Behind the scene, this function calls to a free Python-based server using my own GEE credential. Normally it takes several seconds to retrieve one time series, but as a free cloud service, the Python server only offers 100 seconds of free CPU time per day, with throttling applied. So it may take up to a few mins to get a time series on your end. It may fail due to connection timeout; if so, give it a few tries. If you need to retrieve data for thousands or millions of sites, please contact the author.

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

See Also

[beast](#), [beast.irreg](#), [beast123](#), [minesweeper](#), [tetriss](#)

Examples

```
library(Rbeast)
df = geeLandsat(lon=-80.983877, lat= 40.476882) #if it fails, try a few more times before giving up
print(df)
```

imagestack

Decades of Landsat NDVI time series over a small area in Ohio

Description

imagestack is a LIST containing Landsat-derived NDVI image chips at an Ohio site

Usage

```
data(imagestack)
```

Source

Landsat images courtesy of the U.S. Geological Survey

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

Examples

```
data(imagestack)
imagestack$ndvi # NDVI images collected over the past several deccades
imagestack$datestr # A string vector containg the observation dates of individual ndvi images
```

minesweeper

The Minesweeper game in R

Description

A poor man's implementation of the minesweeper game in R. Yes, you are right: it has nothing to do with time series decomposition, changepoint detection, and time series segmentation. Its only remote connection to Rbeast is that this is a practice script I wrote to learn R graphics for implementing Rbeast.

Usage

```
minesweeper(height=15, width=12, prob=0.1)
```

Arguments

height	integer; number of rows of the mine grid along the vertical direction.
width	integer; number of columns of the mine grid along the horizontal direction.
prob	numeric; a fraction between 0 and 1 to specify the probability of mine occurrence in the mine grid.

Value

Instructions:

- LEFT-click to clear a spot.
- RIGHT-click to flag a spot.
- MIDDLE-click(wheel) a cleared and numbered spot to open neighbor spots, if flagged correctly.
- Click Restart for a new game

Note

An interactive graphics window is needed to run this function correctly. So it won't run in RStudio's plot pane. The function will use the `x11()` or `x11(type='Xlib')` graphic device to open a pop-up window.

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).

3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. ISPRS Journal of Photogrammetry and Remote Sensing, 176, pp.250-261(a beast application paper).

See Also

[beast](#), [beast.irreg](#), [beast123](#), [tetrис](#), [geeLandsat](#)

Examples

```
library(Rbeast)
minesweeper()

# A mine field of size 20x25 with roughly a 15% of the spots being mines.
minesweeper(20,25,0.15)
```

ohio

An irregular Landsat NDVI time series at an Ohio site

Description

ohio is a data.frame object comprising decades of Landsat-observed surface reflectances and NDVI at an Ohio site

Usage

```
data(ohio)
```

Source

Landsat images courtesy of the U.S. Geological Survey

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. Remote Sensing of Environment, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. Remote Sensing of Environment, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).

- Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. ISPRS Journal of Photogrammetry and Remote Sensing, 176, pp.250-261(a beast application paper).

Examples

```
library(Rbeast)
data(ohio) # Landsat surface references and NDVI at a single pixel observed over time
str(ohio)
```

plot.beast

Bayesian changepoint detection and time series decomposition

Description

Plot the result obtained from the beast function.

Usage

```
## S3 method for class 'beast'
plot(
  x,
  index = 1,
  vars = c('st', 's', 'scp', 'sorder', 't', 'tcp', 'torder', 'o', 'ocp', 'error'),
  col = NULL,
  main = "BEAST decomposition and changepoint detection",
  xlab = 'Time',
  ylab = NULL,
  cex.main = 1.2,
  cex.lab = 1,
  relative.heights = NULL,
  interactive = FALSE,
  ...
)
```

Arguments

x a "beast" object returned by `beast`, `beast.irreg`, or `beast123`. It may contain one or many time series.

index an integer (default to 1) or a vector of two integers to specify the index of the time series to plot if x contains results for multiple time series. `index` is always 1 if x has 1 time series. If x is returned by `beast123` with a 2D input, `index` should be a single integer. If x is from `beast123` applied to 3D arrays of time series (e.g., stacked satellite images), `index` can be a linear index or two subscripts to specify the row and column of the pixel/grid.

<code>vars</code>	a vector of strings indicating the elements or variables of <code>x</code> to plot. Possible <code>vars</code> strings include <code>'st'</code> (season plus trend), <code>'s'</code> (season component), <code>'t'</code> (trend component), <code>'o'</code> (outliers), <code>'scp'</code> , <code>'tcp'</code> , <code>'ocp'</code> (occurrence probability of seasonal/trend/outlier changepoint), <code>'sorder'</code> (seasonal harmonic order), <code>'torder'</code> (trend polynomial order), <code>'samp'</code> (amplitude of seasonality), <code>'tslp'</code> (slope of trend), and <code>'error'</code> (remainder).
<code>relative.heights</code>	a numeric vector of the same length as that of <code>vars</code> to specify the relative heights of subplots of individual variables in <code>vars</code> .
<code>col</code>	a string vector of the same length as that of <code>vars</code> to specify the colors of individual subplots associated with <code>vars</code> .
<code>main</code>	a string; the main title.
<code>xlab</code>	a string; the x axis title.
<code>ylab</code>	a string vector of the same length as that of <code>vars</code> to specify the y axis names of individual subplots associated with <code>vars</code>
<code>cex.main</code>	<code>cex</code> for the main title
<code>cex.lab</code>	<code>cex</code> for the axis title
<code>interactive</code>	a bool scalar. If TRUE, an interactive GUI is used for examining individual elements of <code>x</code> .
<code>...</code>	additional parameters to be implemented.

Value

This function creates various plots to demonstrate the results of a beast decomposition. .

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

See Also

[beast](#), [beast.irreg](#), [beast123](#), [minesweeper](#), [tetris](#), [geeLandsat](#)

Examples

```
library(Rbeast)
data(simdata)
result=beast(simdata)
plot(result,1)
plot(result,2)
```

print.beast

Bayesian changepoint detection and time series decomposition

Description

Summarize and print the results obtained from the BEAST time series decomposition and segmentation.

Usage

```
## S3 method for class 'beast'
print(
  x,
  index = 1,
  ...
)
```

Arguments

x a "beast" object returned by [beast](#), [beast.irreg](#), or [beast123](#). It may contain one or many time series.

index an integer (default to 1) or a vector of two integers to specify the index of the time series to print if x contains results for multiple time series. If x has 1 time series, index should be always 1. If x is returned by [beast123](#) applied to a 2D input, index should be a single index. If x is from [beast123](#) applied to 3D arrays of time series (e.g., stacked satellite images), index can be a linear index or two subscripts to specify the row and column of the desired pixel/grid.

... additional parameters to be implemented.

Value

Print a summary of changepoints detected for the seasonal or trend component.

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

See Also

[beast](#), [beast.irreg](#), [beast123](#), [minesweeper](#), [tetris](#), [geeLandsat](#)

Examples

```
library(Rbeast)
data(simdata)
result=beast(simdata)
plot(result,1)
plot(result,2)
```

simdata

Simulated time series to test BEAST

Description

simdata is a 300 x 3 matrix, consisting three time series of length 300. Currently, the three time series are the same. It is used to illustrate BEAST can handle multiple time series at a single function call. of BEAST.

Usage

```
data(simdata)
```

Source

Rbeast v0.0.0

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

Examples

```
library(Rbeast)
data(simdata)
plot(simdata,type='l')

result=beast(simdata)
plot(result,1)
plot(result,2)
plot(result,3)
```

tetris

The Tetris game in R

Description

A poor man's implementation of the Tetris game in R. Yes, you are right again: it has nothing to do with time series decomposition, changepoint detection, and time series segmentation. Its only remote connection to Rbeast is that this is a practice script I wrote to learn R graphics for implementing Rbeast.

Usage

```
tetris(height=25, width=14, speed=0.6)
```

Arguments

height	integer; number of rows of the mine grid along the vertical direction.
width	integer; number of columns of the mine grid along the horizontal direction.
speed	numeric; a time interval between 0.05 and 2 seconds, specifying how fast the tetriminos moves down. The smaller, the faster.

Value

Instructions:

- Left arrow to move left.
- Right arrow to move right.
- Up arrow to rotate.
- Down arrow to speed up.
- Space key to sink to the bottom.

Note

This function works only under the Windows OS not Linux or Mac. An interactive graphics window is needed to run this function correctly. So it won't run in RStudio's plot pane. The function will use the `x11()` or `x11(type='Xlib')` graphic device to open a pop-up window.

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

See Also

[beast](#), [beast.irreg](#), [beast123](#), [minesweeper](#), [geeLandsat](#)

Examples

```
library(Rbeast)

tetris()

# A field of size 20x25 with the blocks moving down every 0.1 second.
tetris(20,25,0.1)
```

tsextract

*Bayesian changepoint detection and time series decomposition***Description**

Extract the result of a single time series from an object of class `beast`

Usage

```
tsextract( x, index = 1 )
```

Arguments

<code>x</code>	a "beast" object returned by <code>beast</code> , <code>beast.irreg</code> , or <code>beast123</code> . It may contain one or many time series.
<code>index</code>	an integer (default to 1) or a vector of two integers to specify the index of the time series to extract if <code>x</code> contains results for multiple time series. If <code>x</code> has 1 time series, <code>index</code> should be always 1. If <code>x</code> is returned by <code>beast123</code> applied to a 2D input, <code>index</code> should be a single index. If <code>x</code> is from <code>beast123</code> applied to 3D arrays of time series (e.g., stacked satellite images), <code>index</code> can be a linear index or two subscripts to specify the row and column of the desired pixel/grid.

Value

A LIST object of the result for the chosen time series, which contains the same field as `x`.

Note

Use this function only to manually and interactively examine individual times series. If the purpose is to loop through `x`, the use of direct indexing is much faster. For example, if `x` is a `beast` object for a 300x200x1000 3D array (row x col x time), use `x$trend$Y[20,40,]` to get the fitted trend at the pixel of row 20 and col 40.

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the `beast` algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in `beast`).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a `beast` application paper).

See Also

[beast](#), [beast.irreg](#), [beast123](#), [minesweeper](#), [tetr](#), [tetr](#), [geeLandsat](#)

Examples

```
library(Rbeast)
data(simdata)
result=beast(simdata)
plot(result,1)
plot(result,2)
```

Yellowstone

30 years' AVHRR NDVI data at a Yellowstone site

Description

Yellowstone is a vector comprising 30 years' AVHRR NDVI data at a Yellowstone site

Usage

```
data(Yellowstone)
```

Source

Rbeast v0.9.1

References

1. Zhao, K., Wulder, M.A., Hu, T., Bright, R., Wu, Q., Qin, H., Li, Y., Toman, E., Mallick, B., Zhang, X. and Brown, M., 2019. Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. *Remote Sensing of Environment*, 232, p.111181 (the beast algorithm paper).
2. Zhao, K., Valle, D., Popescu, S., Zhang, X. and Mallick, B., 2013. Hyperspectral remote sensing of plant biochemistry using Bayesian model averaging with variable and band selection. *Remote Sensing of Environment*, 132, pp.102-119 (the Bayesian MCMC scheme used in beast).
3. Hu, T., Toman, E.M., Chen, G., Shao, G., Zhou, Y., Li, Y., Zhao, K. and Feng, Y., 2021. Mapping fine-scale human disturbances in a working landscape with Landsat time series on Google Earth Engine. *ISPRS Journal of Photogrammetry and Remote Sensing*, 176, pp.250-261(a beast application paper).

Examples

```
library(Rbeast)
data(Yellowstone)
plot(Yellowstone, type='l')
```

```
result=beast(Yellowstone)
plot(result)
```

Index

* misc

- beast, [2](#)
 - beast.irreg, [15](#)
 - beast123, [23](#)
 - CNAchrom11, [41](#)
 - geeLandsat, [42](#)
 - imagestack, [44](#)
 - minesweeper, [45](#)
 - ohio, [46](#)
 - plot.beast, [47](#)
 - print.beast, [49](#)
 - simdata, [50](#)
 - tetris, [51](#)
 - tsextract, [53](#)
 - Yellowstone, [54](#)
-
- BEAST (beast), [2](#)
 - Beast (beast), [2](#)
 - beast, [2](#), [10](#), [16](#), [22](#), [35](#), [43](#), [46–50](#), [52–54](#)
 - beast.123 (beast123), [23](#)
 - BEAST.irreg (beast.irreg), [15](#)
 - beast.irreg, [3](#), [5](#), [6](#), [10](#), [15](#), [35](#), [43](#), [46–50](#), [52–54](#)
 - BEAST123 (beast123), [23](#)
 - beast123, [3](#), [5](#), [6](#), [10](#), [16](#), [22](#), [23](#), [43](#), [46–50](#), [52–54](#)
-
- CNAchrom11, [41](#)
-
- geeLandsat, [10](#), [22](#), [35](#), [42](#), [46](#), [48](#), [50](#), [52](#), [54](#)
-
- imagestack, [44](#)
-
- minesweeper, [10](#), [22](#), [35](#), [43](#), [45](#), [48](#), [50](#), [52](#), [54](#)
-
- Ohio (ohio), [46](#)
 - ohio, [46](#)
-
- plot.beast, [47](#)
 - print.beast, [49](#)
-
- RBEAST (beast), [2](#)
 - Rbeast (beast), [2](#)
 - rbeast (beast), [2](#)
-
- simdata, [50](#)
-
- Tetris (tetris), [51](#)
 - tetris, [10](#), [22](#), [35](#), [43](#), [46](#), [48](#), [50](#), [51](#), [54](#)
 - tsextract, [53](#)
-
- Yellowstone, [54](#)