Package 'DeCAFS'

January 6, 2023

Type Package

Title Detecting Changes in Autocorrelated and Fluctuating Signals

Version 3.3.3

Date 2023-1-6

Maintainer Gaetano Romano <g.romano@lancaster.ac.uk>

Description Detect abrupt changes in time series with local fluctuations as a random walk process and autocorrelated noise as an AR(1) process. See Romano, G., Rigaill, G., Runge, V., Fearnhead, P. (2021) <doi:10.1080/01621459.2021.1909598>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.0), ggplot2, robustbase

LinkingTo Rcpp

NeedsCompilation yes

Depends R (>= 3.5.0)

LazyData true

BugReports https://github.com/gtromano/DeCAFS/issues

RoxygenNote 7.2.0

Author Gaetano Romano [aut, cre],
Guillem Rigaill [aut],
Vincent Runge [aut],
Paul Fearnhead [aut]

Repository CRAN

Date/Publication 2023-01-06 12:10:02 UTC

R topics documented:

`bestParameters` *bestParameters*

Description

iteration of the least square criterion for a grid of the phi parameter

Usage

```
bestParameters(y, nbK = 10, type = "MAD", sdEta = TRUE)
```

Arguments

`y`	A time-series obtained by the dataRWAR function
`nbK`	number of diff k elements to consider
`type`	type of robust variance estimator (MAD, S or Q)
`sdEta`	if sdEta = FALSE there is no random walk

Value

a list with an estimation of the best parameters for Eta2, Nu2 and phi

Examples

```
bestParameters(dataRWAR(10000, sdEta = 0.2, sdNu = 0.1, phi = 0.3,
type = "rand1", nbSeg = 10)$y)
```

`cost` *L2 error estimation*

Description

the least-square value

Usage

```
cost(v, sdEta, sdNu, phi)
```

Arguments

`v`	the estimated variances of the diff k operator
`sdEta`	standard deviation in Random Walk
`sdNu`	standard deviation in AR(1)
`phi`	the autocorrelative AR(1) parameter

Value

the value of the sum of squares

`dataRWAR` *Generate a Random Walk + AR realization*

Description

Generate a Realization from the RWAR model (check the references for further details).

$$y_t = \mu_t + \epsilon_t$$

where

$$\mu_t = \mu_{t-1} + \eta_t + \delta_t, \quad \eta_t \sim N(0, \sigma_\eta^2),\ \delta_t\ \in R$$

and

$$\epsilon_t = \phi\epsilon_{t-1} + \nu_t \quad \nu_t \sim N(0, \sigma_\nu^2)$$

Usage

```
dataRWAR(
  n = 1000,
  sdEta = 0,
  sdNu = 1,
  phi = 0,
  type = c("none", "up", "updown", "rand1"),
  nbSeg = 20,
  jumpSize = 1
)
```

Arguments

`n`	The length of the sequence of observations.
`sdEta`	The standard deviation of the Random Walk Component on the signal drift
`sdNu`	The standard deviation of the Autocorrelated noise
`phi`	The autocorrelation parameter ϕ
`type`	Possible change scenarios for the jump structure (default: `none`)
`nbSeg`	Number of segments
`jumpSize`	Maximum magnitude of a change

Value

A list containing:

`y` the data sequence,

`signal` the underlying signal without the superimposed AR(1) noise,

`changepoints` the changepoint locations

References

Romano, G., Rigaill, G., Runge, V., Fearnhead, P. Detecting Abrupt Changes in the Presence of Local Fluctuations and Autocorrelated Noise. arXiv preprint https://arxiv.org/abs/2005.01379 (2020).

Examples

```
library(ggplot2)
set.seed(42)
Y = dataRWAR(n = 1e3, phi = .5, sdEta = 3, sdNu = 1, jumpSize = 15, type = "updown", nbSeg = 5)
y = Y$y
ggplot(data.frame(t = 1:length(y), y), aes(x = t, y = y)) +
  geom_point() +
  geom_vline(xintercept = Y$changepoints, col = 4,  lty = 3)
```

`dataSinusoidal` *Generating data from a sinusoidal model with changes*

Description

This function generates a sequence of observation from a sinusoidal model with changes. This can be used as an example for model misspecification.

Usage

```
dataSinusoidal(
  n,
  amplitude = 1,
  frequency = 0.001,
  phase = 0,
  sd = 1,
  type = c("none", "up", "updown", "rand1"),
  nbSeg = 20,
  jumpSize = 1
)
```

Arguments

`n`	The length of the sequence of observations.
`amplitude`	The amplitude of the sinusoid
`frequency`	The angular frequency of the sinusoid
`phase`	where the signal starts at time t = 0
`sd`	standard deviation of the noise added on top of the signal
`type`	Possible change scenarios for the jump structure (default: `none`)
`nbSeg`	Number of segments
`jumpSize`	Maximum magnitude of a change

Value

A list containing:

`y` the data sequence,

`signal` the underlying signal without the noise,

`changepoints` the changepoint locations

Examples

```
Y <- dataSinusoidal(
  1e4,
  frequency = 1 / 1e3,
  amplitude = 10,
  type = "updown",
  jumpSize = 4,
  nbSeg = 4
)
res <- DeCAFS(Y$y)
plot(res, col = "grey")
lines(Y$signal, col = "blue", lwd = 2, lty = 2)
abline(v = res$changepoints, col = 2)
abline(v = Y$changepoints, col = 4, lty = 2)
```

DeCAFS *Main DeCAFS algorithm for detecting abrupt changes*

Description

This function implements the DeCAFS algorithm to detect abrupt changes in mean of a univariate data stream in the presence of local fluctuations and auto-correlated noise. It detects the changes under a penalised likelihood model where the data, $y_1, ..., y_n$, is

$$y_t = \mu_t + \epsilon_t$$

with ϵ_t an AR(1) process, and for $t = 2, ..., N$

$$\mu_t = \mu_{t-1} + \eta_t + \delta_t$$

where at time t if we do not have a change then $\delta_t = 0$ and $\eta_t \sim N(0, \sigma_\eta^2)$; whereas if we have a change then $\delta_t \neq 0$ and $\eta_t = 0$. DeCAFS estimates the change by minimising a cost equal to twice the negative log-likelihood of this model, with a penalty β for adding a change. Note that the default DeCAFS behavior will assume the RWAR model, but fit on edge cases is still possible. For instance, should the user wish for DeCAFS to fit an AR model only with a piecewise constant signal, or similarly a model that just assumes random fluctuations in the signal, this can be specified within the initial parameter estimation, by setting the argument: `modelParam = estimateParameters(y, model = "AR")`. Similarly, to allow for negative autocorrelation estimation, set `modelParam = estimateParameters(Y$y, phiLower = -1)`.

Usage

```
DeCAFS(
  data,
  beta = 2 * log(length(data)),
  modelParam = estimateParameters(data, warningMessage = warningMessage),
  penalties = NULL,
  warningMessage = TRUE
)
```

Arguments

`data`	A vector of observations y
`beta`	The l0 penalty. The default one is `2 * log(N)` where `N` is the length of the data.
`modelParam`	A list of 3 initial model parameters: `sdEta`, the SD of the drift (random fluctuations) in the signal, `sdNu`, the SD of the AR(1) noise process, and `phi`, the autocorrelation parameter of the noise process (so the stationary variance of the AR(1) noise process is `sdnu^2 / (1 - phi^2)`. Defaulted to `estimateParameters(data, K = 15)`, to perform automatically estimation of the three. See `estimateParameters()` for more details.

`penalties` Can be used as an alternative to the model parameters, a list of 3 initial penalties: `lambda`, the l2-penalty penalising over the lag-1 of the signal, `gamma`, penalising over the lag-1 of the AR(1) noise process, `phi`, the autocorrelation parameter. These are related to the `modelParam` list by `list(lambda = 1 / sdEta ^ 2, gamma = 1 / sdNu ^ 2, phi = phi)`. Only one argument between `penalties` and `modelParam` should be specified. Defaulted to NULL.

`warningMessage` When `TRUE` prints a message to warn the user that the automatic parameter estimation is employed. Defaults to `TRUE`.

Value

Returns an s3 object of class DeCAFSout where:

`$changepoints` is the vector of change-point locations,

`$signal` is the estimated signal without the auto-correlated noise,

`$costFunction` is the optimal cost in form of piecewise quadratics at the end of the sequence,

`$estimatedParameters` is a list of parameters estimates (if estimated, otherwise simply the initial `modelParam` input),

`$data` is the sequence of observations.

References

Romano, G., Rigaill, G., Runge, V., Fearnhead, P. (2021). Detecting Abrupt Changes in the Presence of Local Fluctuations and Autocorrelated Noise. Journal of the American Statistical Association. doi:10.1080/01621459.2021.1909598.

Examples

```
library(ggplot2)
set.seed(42)
Y <- dataRWAR(n = 1e3, phi = .5, sdEta = 1, sdNu = 3, jumpSize = 15, type = "updown", nbSeg = 5)
y <- Y$y
res = DeCAFS(y)
ggplot(data.frame(t = 1:length(y), y), aes(x = t, y = y)) +
  geom_point() +
  geom_vline(xintercept = res$changepoints, color = "red") +
  geom_vline(xintercept = Y$changepoints, col = "blue",  lty = 3)
```

`estimateParameters` *Estimate parameter in the Random Walk Autoregressive model*

Description

This function perform robust estimation of parameters in the Random Walk plus Autoregressive model using a method of moments estimator. To model the time-dependency DeCAFS relies on three parameters. These are `sdEta`, the standard deviation of the drift (random fluctuations) in the signal, modeled as a Random Walk process, `sdNu`, the standard deviation of the AR(1) noise process, and `phi`, the autocorrelation parameter of the noise process. The final estimation of the change locations is affected by the l0 penalty beta and the estimation of the process by those three initial parameters. Therefore, the choice of penalties for DeCAFS is important: where possible investigate resulting segmentations. Should the algorithm return a misspecified estimation of the signal, it might be good to constrain the estimation of the parameters to an edge case. This can be done through the argument `model`. Alternatively, one could employ a range of penalties or tune these on training data. To manually specify different penalties, see `DeCAFS()` documentation. If unsure of which model is the most suited for a given sequence, see `guidedModelSelection()` for guided model selection.

Usage

```
estimateParameters(
  y,
  model = c("RWAR", "AR", "RW"),
  K = 15,
  phiLower = 0,
  phiUpper = 0.999,
  sdEtaUpper = Inf,
  sdNuUpper = Inf,
  warningMessage = FALSE
)
```

Arguments

`y`	A vector of observations
`model`	Constrain estimation to an edge case of the RWAR model. Defaults to "RWAR". To fit an AR model only with a piece-wise constant signal, specify "AR". To fit a a random walk plus noise, specify "RW".
`K`	The number of K-lags differences of the data to run the robust estimation over. Default set at 15.
`phiLower`	Smallest value of the autocorrelation parameter. Default set at 0.
`phiUpper`	Highest value of the autocorrelation parameter. Default set at 0.99.
`sdEtaUpper`	Highest value of the RW standard deviation. Default set at Inf
`sdNuUpper`	Highest value of the AR(1) noise standard deviation. Default set at Inf
`warningMessage`	A message to warn the user when the automatic parameter estimation is employed.

Value

Returns a list of estimates that can be employed as an argument for parameter `modelParam` to run `DeCAFS()`. Those are:

`sdEta` the SD of the drift (random fluctuations) in the signal,

`sdNu` the SD of the AR(1) noise process,

`phi` the autocorrelation parameter of the noise process.

Examples

```
set.seed(42)
y <- dataRWAR(n = 1e3, phi = .5, sdEta = 1, sdNu = 3,  jumpSize = 15, type = "updown", nbSeg = 5)$y
estimateParameters(y)
```

`estimVar` *Variance estimation for diff k operators*

Description

Estimation of the variances for the diff k operator k = 1 to nbK

Usage

```
estimVar(y, nbK = 10, type = "MAD")
```

Arguments

`y`	A time-series obtained by the dataRWAR function
`nbK`	number of diff k elements to consider
`type`	type of robust variance estimator (MAD, S or Q)

Value

the vector varEst of estimated variances

Examples

```
estimVar(dataRWAR(1000, sdEta = 0.1, sdNu = 0.1, phi = 0.3, type = "rand1",  nbSeg = 10)$y)
```

`evalEtaNu` *RW and AR(1) variance estimations with fixed AR(1) parameter*

Description

Evaluation of the variances Eta2 and Nu2

Usage

```
evalEtaNu(v, phi, sdEta = TRUE)
```

Arguments

`v`	the estimated variances of the diff k operator
`phi`	the autocorrelative AR(1) parameter
`sdEta`	if sdEta = FALSE there is no random walk

Value

a list with an estimation of the variances Eta2 and Nu2

`guidedModelSelection` *Guided Model Selection*

Description

This function aids the user in selecting an appropriate model for a given sequence of observations. The function goes an interactive visualization of different model fits for different choices of initial parameter estimators and l0 penalties (`beta`). At the end, a call to the DeCAFS function is printed, while a DeCAFS wrapper is provvided.

Usage

```
guidedModelSelection(data)
```

Arguments

`data`	A vector of observations y

Value

A function, being a wrapper of DeCAFS with the selected parameter estimators.

Examples

```
## Not run:
y <- dataRWAR(1000, sdEta = 1, sdNu = 4, phi = .4, nbSeg = 4, jumpSize = 20, type = "updown")$y
DeCAFSWrapper <- guidedModelSelection(y)

## End(Not run)
```

`oilWell` *Rock structure data from an oil well*

Description

This data comes from lowering a probe into a bore-hole, and taking measurements of the rock structure as the probe is lowered. As the probe moves from one rock strata to another we expect to see an abrupt change in the signal from the measurements.

Usage

```
oilWell
```

Format

A numeric vector of 4050 obervations

Source

Ruanaidh, Joseph JK O., and William J. Fitzgerald. Numerical Bayesian methods applied to signal processing. Springer Science & Business Media, 2012. doi:10.1007/9781461207177

Examples

```
# removing outliers
n = length(oilWell)
h = 32
med = rep(NA, n)
for (i in 1:n) {
  index = max(1, i - h):min(n, i + h)
  med[i] = median(oilWell[index])
}
residual = (oilWell - med)

y = oilWell[abs(residual) < 8000]
sigma = sqrt(var(residual[abs(residual) < 8000]))

# running DeCAFS
res <- DeCAFS(y/sigma)
plot(res, xlab = "time", ylab = "y", type = "l")
abline(v = res$changepoints, col = 4, lty = 3)
```

plot.DeCAFSout *DeCAFS Plotting*

Description

DeCAFS output plotting method.

Usage

```
## S3 method for class 'DeCAFSout'
plot(x, ...)
```

Arguments

`x`	the output object from a DeCAFS call
`...`	Additional graphical parameters to be passed down to the plot function

Value

An R plot

Examples

```
set.seed(42)
Y <- dataRWAR(n = 1e3, phi = .5, sdEta = 1, sdNu = 3,  jumpSize = 15, type = "updown", nbSeg = 5)
res = DeCAFS(Y$y)
plot(res, type = "l")
```

scenarioGenerator *Generate a piecewise constant signal of a given length*

Description

Generate a piecewise constant signal of a given length

Usage

```
scenarioGenerator(
  n,
  type = c("none", "up", "updown", "rand1"),
  nbSeg = 20,
  jumpSize = 1
)
```

Arguments

`n`	The length of the sequence of observations.
`type`	Possible change scenarios for the jump structure
`nbSeg`	Number of segments
`jumpSize`	Maximum magnitude of a change

Value

a sequence of N values for the piecewise constant signal

Examples

```
scenarioGenerator(1e3, "rand1")
```

Index