

Package: anomalous (via r-universe)

October 22, 2026

Type Package

Title Anomaly Detection using the CAPA and PELT Algorithms

Version 0.0.4.2

Date 2024-11-21

Description Implimentations of the univariate CAPA

<[doi:10.1002/sam.11586](https://doi.org/10.1002/sam.11586)> and PELT

<[doi:10.1080/01621459.2012.737745](https://doi.org/10.1080/01621459.2012.737745)> algoithms along with
various cost functions for different distributions and models.

The modular design, using R6 classes, favour ease of extension
(for example user written cost functions) over the performance
of other implimentations (e.g.

<[doi:10.32614/CRAN.package.changepoint](https://doi.org/10.32614/CRAN.package.changepoint)>,

<[doi:10.32614/CRAN.package.anomaly](https://doi.org/10.32614/CRAN.package.anomaly)>).

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Suggests tinytest, robustbase, knitr, markdown

Imports R6, Rcpp

LinkingTo Rcpp

VignetteBuilder knitr

Depends R (>= 4.0.0)

Language en-GB

BugReports <https://github.com/waternumbers/anomalous/issues>

URL <https://github.com/waternumbers/anomalous>,

<https://waternumbers.github.io/anomalous/>

Repository <https://waternumbers.r-universe.dev>

Date/Publication 2024-12-13 13:43:20 UTC

RemoteUrl <https://github.com/waternumbers/anomalous>

RemoteRef HEAD

RemoteSha fd9c8daf676eab252113d14aac2474ffbdb5be21

Contents

capa	2
categoricalCost	3
coef.anomalous_partition	5
collective_anomalies	5
crops	6
gaussCost	7
gaussRegCost	11
ladCost	15
Lai2005fig4	17
localRegCost	17
machinetemp	19
multinomialCost	19
param	21
partition	21
pelt	22
plot.anomalous_partition	23
point_anomalies	23
poisCost	24
rankCost	26
sim.data	27
summary.anomalous_partition	28
wind	28
Index	31

capa	<i>An R implimentation of the segmented search algorithm</i> <i>pelt algorithm</i>
------	---

Description

An R implimentation of the segmented search algorithm
pelt algorithm

Usage

capa(part, fCost, prune = TRUE, verbose = FALSE)

Arguments

- | | |
|---------|---------------------------------|
| part | the starting partition |
| fCost | the cost function |
| prune | logical, should pruning be used |
| verbose | logical, print out progress |

Details

Basic R implimentation of pelt - not efficent

Value

the optimal partition

categoricalCost	<i>R6 class for the categorical distribution</i>
-----------------	--

Description

Cost functions for the multinomial distribution

Details

Collective anomalies are represented as changes to the expected proportions. Time varying expected proportions are currently not handled.

Methods**Public methods:**

- `categoricalCost$length()`
- `categoricalCost$new()`
- `categoricalCost$baseCost()`
- `categoricalCost$pointCost()`
- `categoricalCost$collectiveCost()`
- `categoricalCost$param()`
- `categoricalCost$clone()`

Method `length()`: Get the length of time series

Usage:

```
categoricalCost$length()
```

Method `new()`: Initialise the cost function

Usage:

```
categoricalCost$new(x, m = rep(1/ncol(x), ncol(x)))
```

Arguments:

`x` integer matrix of observations of 0,1

`m` numeric vector of expected proportions ## need to check row sums are 1 - possibly just use multinomial??

Method `baseCost()`: Compute the non-anomalous cost of a segment

Usage:

```
categoricalCost$baseCost(a, b, pen = 0)
```

Arguments:

`a` start of period

b end of period
 pen penalty cost

Method pointCost(): Compute the point anomaly cost of a time step

Usage:

```
categoricalCost$pointCost(b, pen)
```

Arguments:

b time step
 pen penalty cost

Method collectiveCost(): Compute the anomalous cost of a segment

Usage:

```
categoricalCost$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period
 b end of period
 pen penalty cost
 len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

```
categoricalCost$param(a, b)
```

Arguments:

a start of period
 b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
categoricalCost$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
set.seed(0)
m <- c(1:4)/sum(1:4)
X <- t(rmultinom(100, 1, m))

p <- categoricalCost$new(X,m)
p$baseCost(90,95) ## cost of non-anomalous distribution for x[90:95]
p$pointCost(90,0) ## point anomaly cost for x[90] with 0 penalty
## collective anomaly cost for x[90:95] with penalty of 57 and at least 3 observation
p$collectiveCost(90,95,57,3)
```

coef.anomalous_partition

Coefficient method for an anomalous_partition

Description

Generates coefficients for the anomalous periods in an object with class anomalous_partition

Usage

```
## S3 method for class 'anomalous_partition'
coef(object, ...)
```

Arguments

object	the anomalous_partition
...	optional parameters see details

Details

Generates coefficients for all anomalous periods. Required input cost is a cost function. Optional input t is the time of the partitioning solution to use.

Value

A matrix of parameters whose rows correspond to the rows in the summary output

collective_anomalies *Method for extracting collective anomalies*

Description

Extracts the collective anomaly summaries from the object

Usage

```
collective_anomalies(p, t)
```

Arguments

p	object, such as a partition, from which to extract the point anomalies
t	the end time at which the partition is based

crops

An implimentation of the CROPS algorithm in 1D

Description

An implimentation of the CROPS algorithm in 1D

Usage

```
crops(  
    betaMin,  
    betaMax,  
    fCost,  
    alg = pelt,  
    betaP = Inf,  
    min_length = 2,  
    prune = TRUE,  
    verbose = FALSE,  
    maxIter = 100  
)
```

Arguments

betaMin	lower bound of penalisation window
betaMax	upper bound of penalisation window
fCost	the cost function
alg	algorithm either capa of pelt
betaP	penalty for adding a point anomaly - only for use with capa
min_length	minimum number of values in a collective anomaly
prune	logical, should pruning be used
verbose	logical, print out progress
maxIter	maximum number of algorithm evaluations to perform

Details

This will only work for cost functions where the beta is additive!!!

Value

something...

gaussCost

*R6 class for Univariate Gaussian Cost Functions***Description**

Cost functions for differencing types of univariate gaussian anomalies

Arguments

x	numeric vector or matrix of observations (see details)
m	numeric vector of mean values
s	numeric vector of standard deviation values
point_type	representation of point anomalies as either a change in mean or variance
a	start of period
b	end of period
pen	penalty cost
len	minimum number of observations

Details

Collective anomalies are represented either as changes in mean (gaussMean), variance (gaussVar) or mean and variance (gaussMeanvar). See vignettes for details. If x is a matrix then the values in each row are treated as IID replicate observations.

Methods**Public methods:**

- `gaussCost$length()`
- `gaussCost$new()`
- `gaussCost$baseCost()`
- `gaussCost$pointCost()`
- `gaussCost$clone()`

Method `length()`: Get the length of time series

Usage:

```
gaussCost$length()
```

Method `new()`: Initialise the cost function

Usage:

```
gaussCost$new(x, m = 0, s = 1, point_type = c("var", "mean"))
```

Arguments:

x numeric vector of observations
m numeric vector of mean values

`s` numeric vector of standard deviation values
`point_type` representation of point anomalies as either a change in mean or variance

Method `baseCost()`: Compute the non-anomalous cost of a segment

Usage:

```
gaussCost$baseCost(a, b, pen = 0)
```

Arguments:

`a` start of period

`b` end of period

`pen` penalty cost

Method `pointCost()`: Compute the point anomaly cost of a time step

Usage:

```
gaussCost$pointCost(b, pen)
```

Arguments:

`b` time step

`pen` penalty cost

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
gaussCost$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Super class

```
anomalous::gaussCost -> gaussMean
```

Methods

Public methods:

- `gaussMean$collectiveCost()`
- `gaussMean$param()`
- `gaussMean$clone()`

Method `collectiveCost()`: Compute the anomalous cost of a segment

Usage:

```
gaussMean$collectiveCost(a, b, pen, len)
```

Arguments:

`a` start of period

`b` end of period

`pen` penalty cost

`len` minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

gaussMean\$param(a, b)

Arguments:

a start of period

b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

gaussMean\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Super class

anomalous::gaussCost -> gaussVar

Methods

Public methods:

- gaussVar\$collectiveCost()
- gaussVar\$param()
- gaussVar\$clone()

Method collectiveCost(): Compute the anomalous cost of a segment

Usage:

gaussVar\$collectiveCost(a, b, pen, len)

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

gaussVar\$param(a, b)

Arguments:

a start of period

b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

gaussVar\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Super class

```
anomalous::gaussCost -> gaussMeanVar
```

Methods**Public methods:**

- `gaussMeanVar$collectiveCost()`
- `gaussMeanVar$param()`
- `gaussMeanVar$clone()`

Method `collectiveCost()`: Compute the non-anomalous cost of a segment

Usage:

```
gaussMeanVar$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method `param()`: Compute parameters of a segment if anomalous

Usage:

```
gaussMeanVar$param(a, b)
```

Arguments:

a start of period

b end of period

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
gaussMeanVar$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
set.seed(0)
m <- runif(100)
s <- pmax(1e-4, runif(100))
x <- rnorm(100, m, s) ## example data

gM <- gaussMean$new(x, m, s) ## anomalies are changes in mean
gM$baseCost(90, 95) ## cost of non-anomalous distribution for x[90:95]
gM$pointCost(90, 0) ## point anomaly cost for x[90] with 0 penalty
## collective anomaly cost for x[90:95] with penalty of 57 and at least 3 observation
gM$collectiveCost(90, 95, 57, 3)
```

```

gV <- gaussVar$new(x,m,s) ## anomalies are changes in variance
gV$baseCost(90,95) ## cost of non-anomalous distribution for x[90:95]
gV$pointCost(90,0) ## point anomaly cost for x[90] with 0 penalty
## collective anomaly cost for x[90:95] with penalty of 57 and at least 3 observation
gV$collectiveCost(90,95,57,3)

gMV <- gaussMeanVar$new(x,m,s) ## anomalies are changes in mean and variance
gMV$baseCost(90,95) ## cost of non-anomalous distribution for x[90:95]
gMV$pointCost(90,0) ## point anomaly cost for x[90] with 0 penalty
## collective anomaly cost for x[90:95] with penalty of 57 and at least 3 observation
gMV$collectiveCost(90,95,57,3)

```

gaussRegCost

R6 class for Univariate Gaussian Cost Functions

Description

Cost functions for differencing types of univariate gaussian anomalies

Details

Collective anomalies are represented either as changes in parameters describing the mean (gaussRegMean), variance (gaussRegVar) or mean and variance (gaussRegMeanVar). Changes in variance are represented as a scaling parameter, not changes to the covariance. See vignettes for details.

Each element of the input `x` should be a list containing a vector of observations `y` and corresponding design matrix `X`. Optionally it can also include a vector of parameter `m` and covariance matrix `S`.

Methods

Public methods:

- `gaussRegCost$length()`
- `gaussRegCost$new()`
- `gaussRegCost$baseCost()`
- `gaussRegCost$pointCost()`
- `gaussRegCost$clone()`

Method `length()`: Get the length of time series

Usage:

```
gaussRegCost$length()
```

Method `new()`: Initialise the cost function

Usage:

```
gaussRegCost$new(x, non_neg = FALSE)
```

Arguments:

`x` a list of regressions (see details)

non_neg should only non-negative parameter solutions be considered

Method baseCost(): Compute the non-anomalous cost of a segment

Usage:

```
gaussRegCost$baseCost(a, b, pen = 0)
```

Arguments:

a start of period

b end of period

pen penalty cost

Method pointCost(): Compute the point anomaly cost of a time step

Usage:

```
gaussRegCost$pointCost(b, pen)
```

Arguments:

b time step

pen penalty cost

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
gaussRegCost$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Super class

```
anomalous::gaussRegCost -> gaussRegMean
```

Methods

Public methods:

- gaussRegMean\$collectiveCost()
- gaussRegMean\$param()
- gaussRegMean\$clone()

Method collectiveCost(): Compute the anomalous cost of a segment

Usage:

```
gaussRegMean$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

gaussRegMean\$param(a, b)

Arguments:

a start of period

b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

gaussRegMean\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Super class

anomalous::gaussRegCost -> gaussRegVar

Methods

Public methods:

- gaussRegVar\$collectiveCost()
- gaussRegVar\$param()
- gaussRegVar\$clone()

Method collectiveCost(): Compute the anomalous cost of a segment

Usage:

gaussRegVar\$collectiveCost(a, b, pen, len)

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

gaussRegVar\$param(a, b)

Arguments:

a start of period

b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

gaussRegVar\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Super class

`anomalous::gaussRegCost -> gaussRegMeanVar`

Methods**Public methods:**

- `gaussRegMeanVar$collectiveCost()`
- `gaussRegMeanVar$param()`
- `gaussRegMeanVar$clone()`

Method `collectiveCost()`: Compute the anomalous cost of a segment

Usage:

```
gaussRegMeanVar$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method `param()`: Compute parameters of a segment if anomalous

Usage:

```
gaussRegMeanVar$param(a, b)
```

Arguments:

a start of period

b end of period

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
gaussRegMeanVar$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## simple test
set.seed(10)
x <- list()
n <- 120
for(ii in 1:48){

  if(ii < 10){ theta = c(1,0); sigma <- 0.1 }
  if(ii >= 10 & ii <12){ theta <- c(10,0); sigma <- 2}
  if(ii >= 12 & ii < 44){ theta <- c(5,1); sigma <- 2}
  if(ii >= 44 ){ theta <- c(1,0); sigma <- 0.1}
```

```

X <- cbind(rep(1,n),runif(n,ii-1,ii))
y <- rnorm(n, X%%theta, sigma)
x[[ii]] <- list(y=y,X=X)
}

fCost <- gaussRegMeanVar$new(x)
p <- partition(4*log(sum(sapply(y,length))),NA,2)
res <- pelt(p,fCost)

```

ladCost

*R6 class for LAD Cost Functions***Description**

Cost functions for the Least Absolute Deviation from a given quantile

Details

this is a very niaive and slow implimentation

Methods**Public methods:**

- `ladCost$length()`
- `ladCost$new()`
- `ladCost$baseCost()`
- `ladCost$pointCost()`
- `ladCost$collectiveCost()`
- `ladCost$param()`
- `ladCost$clone()`

Method `length()`: Get the length of time series

Usage:

`ladCost$length()`

Method `new()`: Initialise the cost function

Usage:

`ladCost$new(x, m = 0, tau = 0.5)`

Arguments:

`x` numeric vector of observations

`m` expected value of `x`

`tau` the quantile

Method `baseCost()`: Compute the non-anomalous cost of a segment

Usage:

```
ladCost$baseCost(a, b, pen = 0)
```

Arguments:

a start of period

b end of period

pen penalty cost

Method pointCost(): Compute the point anomaly cost of a time step

Usage:

```
ladCost$pointCost(a, pen)
```

Arguments:

a time step

pen penalty cost

Method collectiveCost(): Compute the anomalous cost of a segment

Usage:

```
ladCost$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

```
ladCost$param(a, b)
```

Arguments:

a start of period

b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ladCost$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
set.seed(0)
m <- runif(100)
x <- rnorm(100,m)

p <- ladCost$new(x,m,0.5)
p$baseCost(90,95) ## cost of non-anomalous distribution for x[90:95]
p$pointCost(90,0) ## point anomaly cost for x[90] with 0 penalty
## collective anomaly cost for x[90:95] with penalty of 57 and at least 3 observation
p$collectiveCost(90,95,57,3)
```

Lai2005fig4	<i>Normalized glioblastoma profile for an excerpt of chromosome 7, the EGFR locus.</i>
-------------	--

Description

This dataset is taken from Lai W, Johnson MJ, Kucherlapati R, Park PJ, Bioinformatics , 2005. The paper states that the original source of the data is from Bredel et al. (2005). The data is an excerpt of chromosome 7 in GBM29 from 40 to 65 Mb.

This version of the data is a copy of that in the changepoint package.

Usage

```
data(Lai2005fig4)
```

Format

A matrix of dimensions 193 x 5. The columns are Spot, CH, POS.start, POS.end, GBM31.

Source

http://compbio.med.harvard.edu/Supplements/Bioinformatics05b/Profiles/Chrom_7_from40_to65Mb_GBM29.xls

localRegCost	<i>R6 class for local regression Cost Functions</i>
--------------	---

Description

VERY experimental - do not use

Methods

Public methods:

- `localRegCost$length()`
- `localRegCost$new()`
- `localRegCost$baseCost()`
- `localRegCost$pointCost()`
- `localRegCost$collectiveCost()`
- `localRegCost$param()`
- `localRegCost$clone()`

Method `length()`: Get the length of time series

Usage:

```
localRegCost$length()
```

Method new(): Initialise the cost function

Usage:

```
localRegCost$new(x, family = c("gaussian", "symmetric"))
```

Arguments:

x observations as for gauss_reg
family for fitting

Method baseCost(): Compute the non-anomalous cost of a segment

Usage:

```
localRegCost$baseCost(a, b, pen = 0)
```

Arguments:

a start of period
b end of period
pen penalty cost

Method pointCost(): Compute the point anomaly cost of a time step

Usage:

```
localRegCost$pointCost(b, pen)
```

Arguments:

b time step
pen penalty cost

Method collectiveCost(): Compute the non-anomalous cost of a segment

Usage:

```
localRegCost$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period
b end of period
pen penalty cost
len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

```
localRegCost$param(a, b)
```

Arguments:

a start of period
b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
localRegCost$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

machinetemp	<i>Machine temperature data.</i>
-------------	----------------------------------

Description

Temperature sensor data of an internal component of a large, industrial machine. The data contains three known anomalies. The first anomaly is a planned shutdown of the machine. The second anomaly is difficult to detect and directly led to the third anomaly, a catastrophic failure of the machine. The data consists of 22695 observations of machine temperature recorded at 5 minute intervals along with the date and time of the measurement. The data was obtained from the Numenta Anomaly Benchmark, which can be found at <https://github.com/numenta/NAB>.

Usage

```
data(machinetemp)
```

Format

A dataframe with 22695 rows and 2 columns. The first column contains the date and time of the temperature measurement. The second column contains the machine temperature.

multinomialCost	<i>R6 class for the multinomial distribution</i>
-----------------	--

Description

Cost functions for the multinomial distribution

Details

Collective anomalies are represented as changes to the expected proportions. Time varying expected proportions are currently not handled.

Methods

Public methods:

- `multinomialCost$length()`
- `multinomialCost$new()`
- `multinomialCost$baseCost()`
- `multinomialCost$pointCost()`
- `multinomialCost$collectiveCost()`
- `multinomialCost$param()`
- `multinomialCost$clone()`

Method `length()`: Get the length of time series

Usage:

```
multinomialCost$length()
```

Method new(): Initialise the cost function

Usage:

```
multinomialCost$new(x, m = rep(1/ncol(x), ncol(x)))
```

Arguments:

x integer matrix of observations

m numeric vector of expected proportions

Method baseCost(): Compute the non-anomalous cost of a segment

Usage:

```
multinomialCost$baseCost(a, b, pen = 0)
```

Arguments:

a start of period

b end of period

pen penalty cost

Method pointCost(): Compute the point anomaly cost of a time step

Usage:

```
multinomialCost$pointCost(b, pen)
```

Arguments:

b time step

pen penalty cost

Method collectiveCost(): Compute the anomalous cost of a segment

Usage:

```
multinomialCost$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

```
multinomialCost$param(a, b)
```

Arguments:

a start of period

b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
multinomialCost$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```

set.seed(0)
m <- c(1:4)/sum(1:4)
X <- t(rmultinom(100, 144, m))

p <- multinomialCost$new(X,m)
p$baseCost(90,95) ## cost of non-anomalous distribution for x[90:95]
p$pointCost(90,0) ## point anomaly cost for x[90] with 0 penalty
## collective anomaly cost for x[90:95] with penalty of 57 and at least 3 observation
p$collectiveCost(90,95,57,3)

```

param	<i>Get the parameters for a partitioning result</i>
-------	---

Description

Get the parameters for a partitioning result

Usage

```
param(res, fCost)
```

Arguments

res	the result of a partitioning algorithm
fCost	the cost function

Details

Not yet implimented for all cost functions

Value

list of parameters

partition	<i>Partion record for the data</i>
-----------	------------------------------------

Description

A partition records the seperation of the data generated by the pelt or capa methods

Usage

```
partition(beta, betaP, min_length)
```

Arguments

beta	penalty for a new collective anomaly
betaP	penalty for a new point anomaly
min_length	shortest length of a collective anomaly

Examples

```
p <- partition(3,4,2)
```

pelt	<i>An R implimentation of the segmented search algorithm</i>
------	--

Description

An R implimentation of the segmented search algorithm

Usage

```
pelt(part, fCost, prune = TRUE, verbose = FALSE)
```

Arguments

part	the starting partition
fCost	the cost function
prune	logical, should pruning be used
verbose	logical, print out progress

Details

Basic R implimentation of pelt - not efficent

Value

the optimal partition

plot.anomalous_partition

Plot method for an amomalous_partition

Description

Provides a summary plot of an object with class amomalous_partition

Usage

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

Arguments

x	the anomalous_partition
...	optional parameters see details

Details

If providing t a time the results prodiced are as though the analysis was stopped then. The optional inputs xx and yy allow for data to be displayed under the shaded anomaly/changepoint areas. If no data is provided xx = 1:t and yy is the cost of the periods the timestep is in.

Value

Produces a plot

point_anomalies

Method for extracting point anomalies

Description

Extracts the point anomaly summaries from the object

Usage

```
point_anomalies(p, t)
```

Arguments

p	object, such as a partition, from which to extract the point anomalies
t	the end time at which the partition is based

poisCost

R6 class for Univariate Poisson Cost Functions

Description

Cost functions for the univariate Poission distribution

Details

Collective anomalies are represented as multiplicative changes in rate

Methods

Public methods:

- `poisCost$length()`
- `poisCost$new()`
- `poisCost$baseCost()`
- `poisCost$pointCost()`
- `poisCost$collectiveCost()`
- `poisCost$param()`
- `poisCost$clone()`

Method `length()`: Get the length of time series

Usage:

```
poisCost$length()
```

Method `new()`: Initialise the cost function

Usage:

```
poisCost$new(x, rate = 1)
```

Arguments:

x numeric vector of observations

rate numeric vector of rate parameters

Method `baseCost()`: Compute the non-anomalous cost of a segment

Usage:

```
poisCost$baseCost(a, b, pen = 0)
```

Arguments:

a start of period

b end of period

pen penalty cost

Method `pointCost()`: Compute the point anomaly cost of a time step

Usage:

```
poisCost$pointCost(b, pen)
```

Arguments:

b time step

pen penalty cost

Method collectiveCost(): Compute the anomalous cost of a segment

Usage:

```
poisCost$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

```
poisCost$param(a, b)
```

Arguments:

a start of period

b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
poisCost$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
set.seed(0)
r <- 8 + runif(100)*2
x <- rpois(100,lambda = r)

p <- poisCost$new(x,r)
p$baseCost(90,95) ## cost of non-anomalous distribution for x[90:95]
p$pointCost(90,0) ## point anomaly cost for x[90] with 0 penalty
## collective anomaly cost for x[90:95] with penalty of 57 and at least 3 observation
p$collectiveCost(90,95,57,3)
```

rankCost

R6 class for rank based anomalies

Description

This is VERY developmental - do not use

Methods

Public methods:

- `rankCost$length()`
- `rankCost$new()`
- `rankCost$baseCost()`
- `rankCost$pointCost()`
- `rankCost$collectiveCost()`
- `rankCost$param()`
- `rankCost$clone()`

Method `length()`: Get the length of time series

Usage:

`rankCost$length()`

Method `new()`: Initialise the cost function

Usage:

`rankCost$new(x, m = 0)`

Arguments:

x numeric matrix of observations

m numeric vector or matrix of mean values

Method `baseCost()`: Compute the non-anomalous cost of a segment

Usage:

`rankCost$baseCost(a, b, pen = 0)`

Arguments:

a start of period

b end of period

pen penalty cost

Method `pointCost()`: Compute the point anomaly cost of a time step

Usage:

`rankCost$pointCost(b, pen)`

Arguments:

b time step

pen penalty cost

Method collectiveCost(): Compute the non-anomalous cost of a segment

Usage:

```
rankCost$collectiveCost(a, b, pen, len)
```

Arguments:

a start of period

b end of period

pen penalty cost

len minimum number of observations

Method param(): Compute parameters of a segment if anomalous

Usage:

```
rankCost$param(a, b)
```

Arguments:

a start of period

b end of period

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
rankCost$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
set.seed(0)
m <- runif(100)
s <- pmax(1e-4, runif(100))
x <- rnorm(100, m, s) ## example data
```

sim.data

Simulated data.

Description

A simulated data set for use in the examples and vignettes. The data consists of 500 observations on 20 variates drawn from the standard normal distribution. Within the data there are three multivariate anomalies of length 15 located at $t=100$, $t=200$, and $t=300$ for which the mean changes from 0 to 2. The anomalies affect variates 1 to 8, 1 to 12 and 1 to 16 respectively.

Usage

```
data(simulated)
```

Format

A matrix with 500 rows and 40 columns.

```
summary.anomalous_partition
```

Summary method for an amomalous_partition

Description

Provides a summary of an object with class `anomalous_partition`

Usage

```
## S3 method for class 'anomalous_partition'
summary(object, ...)
```

Arguments

<code>object</code>	the <code>anomalous_partition</code>
<code>...</code>	optional parameters see details

Details

If providing `t` a time the results produced are as though the analysis was stopped then.

Value

A `data.frame` summarising the partitions

<code>wind</code>	<i>Ireland wind data, 1961-1978</i>
-------------------	-------------------------------------

Description

Daily average wind speeds for 1961-1978 at 12 synoptic meteorological stations in the Republic of Ireland (Haslett and raftery 1989). Wind speeds are in knots (1 knot = 0.5418 m/s), at each of the stations in the order given in Fig.4 of Haslett and Raftery (1989, see below)

This data is a copy of that contained within the `gstat` package

Usage

```
data(wind)
```

Format

data.frame wind contains the following columns:

year year, minus 1900

month month (number) of the year

day day

RPT average wind speed in knots at station RPT

VAL average wind speed in knots at station VAL

ROS average wind speed in knots at station ROS

KIL average wind speed in knots at station KIL

SHA average wind speed in knots at station SHA

BIR average wind speed in knots at station BIR

DUB average wind speed in knots at station DUB

CLA average wind speed in knots at station CLA

MUL average wind speed in knots at station MUL

CLO average wind speed in knots at station CLO

BEL average wind speed in knots at station BEL

MAL average wind speed in knots at station MAL

data.frame wind.loc contains the following columns:

Station Station name

Code Station code

Latitude Latitude, in DMS, see examples below

Longitude Longitude, in DMS, see examples below

MeanWind mean wind for each station, metres per second

Note

This data set comes with the following message: “Be aware that the dataset is 532494 bytes long (thats over half a Megabyte). Please be sure you want the data before you request it.” The data were obtained on Oct 12, 2008, from: <http://www.stat.washington.edu/raftery/software.html> The data are also available from statlib. Locations of 11 of the stations (ROS, Rosslare has been thrown out because it fits poorly the spatial correlations of the other stations) were obtained from: <http://www.stat.washington.edu/research/reports/2005/tr475.pdf> Roslare lat/lon was obtained from google maps, location Roslare. The mean wind value for Roslare comes from Fig. 1 in the original paper. Haslett and Raftery proposed to use a sqrt-transform to stabilize the variance.

Author(s)

Adrian Raftery; imported to R by Edzer Pebesma

References

These data were analyzed in detail in the following article:

Haslett, J. and Raftery, A. E. (1989). Space-time Modelling with Long-memory Dependence: Assessing Ireland's Wind Power Resource (with Discussion). *Applied Statistics* 38, 1-50. and in many later papers on space-time analysis, for example: Tilmann Gneiting, Marc G. Genton, Peter Guttorp: Geostatistical Space-Time Models, Stationarity, Separability and Full symmetry. Ch. 4 in: B. Finkenstaedt, L. Held, V. Isham, *Statistical Methods for Spatio-Temporal Systems*.

Examples

```
data(wind)
summary(wind)
```

Index

* datasets

Lai2005fig4, [17](#)
machinetemp, [19](#)
sim.data, [27](#)
wind, [28](#)

anomalous::gaussCost, [8–10](#)
anomalous::gaussRegCost, [12–14](#)

capa, [2](#)
categoricalCost, [3](#)
coef.anomalous_partition, [5](#)
collective_anomalies, [5](#)
crops, [6](#)

gaussCost, [7](#)
gaussMean (gaussCost), [7](#)
gaussMeanVar (gaussCost), [7](#)
gaussRegCost, [11](#)
gaussRegMean (gaussRegCost), [11](#)
gaussRegMeanVar (gaussRegCost), [11](#)
gaussRegVar (gaussRegCost), [11](#)
gaussVar (gaussCost), [7](#)

ladCost, [15](#)
Lai2005fig4, [17](#)
localRegCost, [17](#)

machinetemp, [19](#)
multinomialCost, [19](#)

param, [21](#)
partition, [21](#)
pelt, [22](#)
plot.anomalous_partition, [23](#)
point_anomalies, [23](#)
poisCost, [24](#)

rankCost, [26](#)

sim.data, [27](#)

summary.anomalous_partition, [28](#)

wind, [28](#)