Package ‘gfpop’

April 6, 2020

Type Package

Title Graph-constrained Functional Pruning Optimal Partitioning

Version 0.2.2

Maintainer Vincent Runge <vincent.runge@univ-evry.fr>

Description Penalized parametric changepoint detection by functional pruning dynamic programming algorithm.
The successive means are constrained using a graph structure with edges of type null, up, down, std or abs. To each edge we can associate some additional properties: a minimal gap size, a penalty, some robust parameters (K,a). The user can also constrain the inferred means to lie between some minimal and maximal values. Data is modelized by a quadratic cost with possible use of a robust loss, biweight and Huber (see edge parameters K and a). Other losses are also available with log-linear representation (variance, poisson, exp) or a log-log representation (negbin).

License GPL-3

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.18)

SystemRequirements C++11

DevPackage vrunge/fpop

LinkingTo Rcpp

RoxygenNote 6.1.1

Suggests knitr, data.table, testthat, rmarkdown, ggplot2, penaltyLearning

VignetteBuilder knitr

Depends R (>= 2.10)

R topics documented:

`dataGenerator` *Data Generator*

Description

Generating data with a given model = changepoint relative positions + parameters + type of cost + (standard deviation + gamma decay)

Usage

```
dataGenerator(n, changepoints, parameters, type = "mean", sigma = 1,
  gamma = 1, size = 10)
```

Arguments

`n`	number of data points to generate
`changepoints`	vector of position of the changepoints in (0,1] (last element is always 1).
`parameters`	vector of means for the consecutive segments (same length as changepoints)
`type`	a string defining the cost model to use: "mean", "variance", "poisson", "exp", "negbin"
`sigma`	a positive number = the standard deviation of the data
`gamma`	a number between 0 and 1 : the coefficient of the exponential decay (by default = 1 for piecewise constant signals)
`size`	parameter of the rnbinom function

Value

a vector of size n generated by the chosen model

Examples

```
dataGenerator(100, c(0.3, 0.6, 1), c(1, 2, 3))
```

ECG *Electrocardiogram data*

Description

The goal in these data is to detect the QRS complex, as does the Pan-Tompkins algorithm.

Usage

```
data("ECG")
```

Format

A named list of two data.tables: data has 1001 rows and 2 columns, PanTompkins has 24 rows and 3 columns.

References

https://www.robots.ox.ac.uk/~gari/teaching/cdt/A3/readings/ECG/Pan+Tompkins.pdf

Edge *Edge generation*

Description

Edge creation for graph

Usage

```
Edge(state1, state2, type = "null", decay = 1, gap = 0,
  penalty = 0, K = Inf, a = 0)
```

Arguments

`state1`	a string defining the starting state of the edge
`state2`	a string defining the ending state of the edge
`type`	a string equal to "null", "std", "up", "down" or "abs"
`decay`	a nonnegative number to give the strength of the exponential decay into the segment
`gap`	a nonnegative number to constrain the size of the gap in the change of state
`penalty`	a nonnegative number. The penality associated to this state transition
`K`	a positive number. Threshold for the Biweight robust loss
`a`	a positive number. Slope for the Huber robust loss

Value

a one-row dataframe with 9 variables

Examples

```
Edge("Dw", "Up", "up", gap = 1, penalty = 10, K = 3)
```

---

gfpop *Graph-Constrained Functional Pruning Optimal Partitioning*

Description

Functional pruning optimal partitioning with a graph structure to take into account constraints on consecutive segment parameters. The user has to specify the graph he wants to use (see the graph function) and a type of cost funcion. This is the main function of the gfpop package.

Usage

```
gfpop(data, mygraph, type = "mean", weights = NULL, testMode = FALSE)
```

Arguments

`data`	vector of data to segment
`mygraph`	dataframe of class "graph" to constrain the changepoint inference
`type`	a string defining the cost model to use: "mean", "variance", "poisson", "exp", "negbin"
`weights`	vector of weights (positive numbers), same size as data
`testMode`	boolean. False by default. Used to debug the code

Value

a gfpop object = (changepoints, states, forced, parameters, globalCost)

`changepoints` is the vector of changepoints (we give the last element of each segment)

`states` is the vector giving the state of each segment

`forced` is the vector specifying whether the constraints of the graph are active (=1) or not (=0)

`parameters` is the vector of successive parameters of each segment

`globalCost` is a number equal to the total loss: the minimal cost for the optimization problem with all penalty values excluded

graph *Graph generation*

Description

Graph creation using component functions "Edge", "StartEnd" and "Node"

Usage

```
graph(..., type = "empty", decay = 1, gap = 0, penalty = 0,
  K = Inf, a = 0, all.null.edges = FALSE)
```

Arguments

`...`	This is a list of edges definied by functions "Edge", "StartEnd" and "Node"
`type`	a string equal to "std", "isotonic", "updown", "relevant". to build a predefined classic graph
`decay`	a nonnegative number to give the strength of the exponential decay into the segment
`gap`	a nonnegative number to constrain the size of the gap in the change of state
`penalty`	a nonnegative number equals to the common penalty to use for all edges
`K`	a positive number. Threshold for the Biweight robust loss
`a`	a positive number. Slope for the Huber robust loss
`all.null.edges`	a boolean. Add null edges to all nodes automatically

Value

a dataframe with 9 variables (columns are named "state1", "state2", "type", "parameter", "penalty", "K", "a", "min", "max") with additional "graph" class.

Examples

```
UpDownGraph <- graph(type = "updown", gap = 1.3, penalty = 10)
MyGraph <- graph(Edge("Dw","Dw"), Edge("Up","Up"), Edge("Dw","Up","up", gap = 0.5, penalty = 10),
Edge("Up","Dw","down"), StartEnd("Dw","Dw"), Node("Dw",0,1), Node("Up",0,1))
```

itergfpop — *Graph-constrained functional pruning optimal partitioning iterated*

Description

Functional pruning optimal partitioning with a graph structure to take into account constraints on consecutive segment parameters. This is an iterated version of the main gfpop function using a Birgé Massart like penalty

Usage

```
itergfpop(data, mygraph, type = "mean", weights = NULL,
  iter.max = 100, D.init = 1)
```

Arguments

`data`	vector of data to segment
`mygraph`	dataframe of class graph to constrain the changepoint inference
`type`	a string defining the cost model to use: "gauss", "variance", "poisson", "exp", "negbin"
`weights`	vector of weights (positive numbers), same size as data
`iter.max`	maximal number of iteration of the gfpop function
`D.init`	initialisation of the number of segments

Value

a gfpop object = (changepoints, states, forced, parameters, globalCost, Dvect)

`changepoints` is the vector of changepoints (we give the last element of each segment)

`states` is the vector giving the state of each segment

`forced` is the vector specifying whether the constraints of the graph are active (=1) or not (=0)

`parameters` is the vector of successive parameters of each segment

`globalCost` is a number equal to the total loss: the minimal cost for the optimization problem with all penalty values excluded

`Dvect` is a vector of integers. The successive tested D in the Birgé Massart penalty until convergence

`Mono27ac` *A small ChIP-seq data set in which peaks can be found using PeakSegFPOP*

Description

The data come from an H3K27ac ChIP-seq experiment which was aligned to the human reference genome (hg19), aligned read counts were used to produce the coverage data; looking at these data in a genome browser was used to produce the labels.

Usage

```
data("Mono27ac")
```

Format

A list of 2 data.tables: coverage has 4 columns (chrom, chromStart, chromEnd, count=number of aligned reads at each position on chrom:chromStart-chromEnd); labels has 4 columns (chrom, chromStart, chromEnd, annotation=label at chrom:chromStart-chromEnd).

Source

https://github.com/tdhock/feature-learning-benchmark, prob.dir= H3K27ac-H3K4me3_TDHAM_BP/samples/Mono1_H3K
580000

`neuroSpike` *Neuro spike train data*

Description

The goal in these data s to detect the spikes = abrupt up changes in calcium concentration, which correspond to neurons firing in laboratory animals.

Usage

```
data("neuroSpike")
```

Format

A data frame with 20000 observations on the following 3 variables.

`seconds` time of measurement

`calcium` noisy data measured

`AR1` autoregressive model mean, for comparison

Source

Spikefinder challenge, dataset number 7, cell number 13, subset 1:20000, 100 fps, for AR1: 0.98 exp decay parameter, 3.6 penalty parameter.

Node *Node Values*

Description

Constrain the range of values to consider at a node

Usage

```
Node(state = NULL, min = -Inf, max = Inf)
```

Arguments

`state`	a string defining the state to constrain
`min`	minimal value for the inferred parameter
`max`	maximal value for the inferred parameter

Value

a dataframe with 9 variables with only 'state1', 'min' and 'max' defined.

Examples

```
Node(state = "s0", min = 0, max = 2)
```

---

paperGraph *Graphs of our paper in jss*

Description

Graphs of our paper in jss

Usage

```
paperGraph(nb, penalty = 0, decay = 1, gap = 0, oneValue = 0,
  K = Inf, a = 0)
```

Arguments

`nb`	the number of the Figure in paper
`penalty`	the penalty to use for change-point
`decay`	a nonnegative number to give the strength of the exponential decay into the segment
`gap`	a nonnegative number to constrain the size of the gap in the change of state
`oneValue`	the value for parameter when we consider the collective anomalies problem
`K`	a positive number. Threshold for the Biweight robust loss
`a`	a positive number. Slope for the Huber robust loss

Value

a dataframe with 9 variables (columns are named "state1", "state2", "type", "parameter", "penalty", "K", "a", "min", "max") with additional "graph" class.

`plot.gfpop` *plot.gfpop*

Description

Plot the result of the gfpop function with the data-points

Usage

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

Arguments

`x`	a gfpop class object
`...`	Other parameters
`data`	the data from which we get the gfpop result

Value

plot data and the inferred gfpop segments

`profile614chr2` *Labeled DNA copy number profile*

Description

This DNA copy number data set has changepoint labels that are impossible to all predict correctly using the maximum likelihood gaussian changepoint model.

Usage

```
data("profile614chr2")
```

Format

Named list of two elements: probes is a data.table with 153663 noisy logratio observations in which we would like to detect changepoints; labels is a data.table with 8 rows that define regions with 1 or 0 breakpoints.

Source

http://members.cbio.mines-paristech.fr/~thocking/neuroblastoma/test-data-01/614-regions.csv.gz http://members.cbio.mines
paristech.fr/~thocking/neuroblastoma/test-data-01/614-probes.csv.gz

References

Labels created with SegAnnDB software, Hocking et al, Bioinformatics 2014. https://www.ncbi.nlm.nih.gov/pubmed/244930

`sdDiff` *sdDiff*

Description

Three methods to estimate the standard deviation in a time series for change-in-mean problem

Usage

```
sdDiff(x, method = "HALL")
```

Arguments

`x`	vector of datapoint
`method`	Three available methods: "HALL", "MAD" and "SD"

Value

a value equal to the estimated standard deviation

Examples

```
data <- dataGenerator(100, c(0.3, 0.6, 1), c(1, 2, 3), sigma = 2)
sdDiff(data)
```

`StartEnd` *Start and End nodes for the graph*

Description

Defining the beginning and ending states of a graph

Usage

```
StartEnd(start = NULL, end = NULL)
```

Arguments

`start`	a vector of states. The beginning nodes for the changepoint inference
`end`	a vector of states. The ending nodes for the changepoint inference

Value

dataframe with 9 variables with only ‘state1‘ and ‘type‘ = start or end defined.

Examples

```
StartEnd(start = "A", end = c("A","B"))
```

Index