Package: segmentr (via r-universe)

September 13, 2024

Type Package

Title Segment Data Minimizing a Cost Function

Version 0.2.0

Maintainer Thales Mello <thalesmello@gmail.com>

Description Given a cost function provided by the user, this package applies it to a given matrix dataset in order to find change points in the data that minimize the sum of the costs of all the segments. This package provides a handful of algorithms with different time complexities and assumption compromises so the user is able to choose the best one for the problem at hand. The implementation of the segmentation algorithms in this package are based on the paper by Bruno M. de Castro, Florencia Leonardi (2018) <arXiv:1501.01756>. The Berlin weather sample dataset was provided by Deutscher Wetterdienst <https://dwd.de/>. You can find all the references in the Acknowledgments section of this package's repository via the URL below.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports Rcpp (>= 0.12.16), foreach, glue

LinkingTo Rcpp

Suggests testthat, doParallel, knitr, rmarkdown, tidyr, tibble, dplyr, lubridate, magrittr, rdwd, purrr

RoxygenNote 6.1.1

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Language en-US

URL https://github.com/thalesmello/segmentr

Repository https://segmentr-package.r-universe.dev

RemoteUrl https://github.com/segmentr-package/segmentr

RemoteRef HEAD

RemoteSha a75c0affff108c9b29493b96a76a7b21b14389fe

Contents

`auto_penalize` *Penalize a cost function with a guessed penalty function*

Description

Given a dataset, a cost function and penalty parameters on how to penalize big and small segments, this function makes an educated guess on a added penalty for the cost function.

Usage

```
auto_penalize(data, likelihood, cost, big_segment_penalty = 10,
  small_segment_penalty = 10)
```

Arguments

`data` dataset to be segmented by the `segment()` function

`likelihood` function to be maximized using the `segment()` function. It's used to find out the scale of the values in the segment function. Deprecated.

`cost` function to be minimized using the `segment()` function. It's used to find out the scale of the values in the segment function

`big_segment_penalty`
penalty factor for big segments. The bigger it is, the bigger the penalty on big segments. Must be greater than or equal to 1. Penalty on big segments is constant when it's equal to 1. Default: 10

`small_segment_penalty`
penalty factor for small segments. The bigger it is, the bigger the penalty on small segments. Must be greater than or equal to 1. Penalty on small segments is constant when it's equal to 1. Default: 10

Details

This function tries to fit a sum of two exponential functions to values inferred from the dataset and the penalty function. The model for the penalty function we try to fit is in the form:

$$C1exp(s1(x - L/2)) + C2exp(s2(-x + L/2))$$

In the equation, $C1$ and $s1$ are, respectively, a multiplier constant and an exponential scale modifier for small segments, whereas $C2$ and $s2$ are the equivalent ones for big segments. L is the number of columns in the `data` matrix.

Assuming the penalty function to be as such, the parameters are estimated considering the scale of values yielded by the cost function for small and big segments, also taking into account the `big_segment_penalty` and `small_segment_penalty` tuning parameters, which can be used to adjust the effect of the penalty function over big and small segments, respectively.

Value

the likelihood function with the guessed penalty function applied

Examples

```
## Not run:
penalized_cost <- auto_penalize(berlin, multivariate)

## End(Not run)
```

`berlin` *Daily temperatures from weather stations in Berlin*

Description

Contains weather daily weather data from many Deutscher Wetterdienst weather stations in Berlin from the years of 2010 and 2011. Data was obtained using the package rdwd and reformatted to a format appropriate to be used for analysis in this object.

Usage

```
berlin
```

Format

A matrix containing daily temperatures, with each column representing a date and each column representing a weather station in Berlin

rows

columns dates from the years 2010 and 2011 ...

Source

`https://www.dwd.de/DE/Home/home_node.html`

exactalg *Segment data into exact change points*

Description

Find changes points with minimal total cost comparing all possible segment combinations

Usage

```
exactalg(data, cost, likelihood, max_segments = ncol(data),
  allow_parallel = TRUE)
```

Arguments

`data` matrix for which to find the change points

`cost` a function receives the segment matrix as argument and returns a cost for the segment. This function is used to calculate the change points that minimize the total cost. Depending on the algorithm being used, this function is likely to be executed many times, in which case it's also likely to be the bottleneck of the function execution, so it's good for this function to have a fast implementation.

`likelihood` deprecated: use cost instead. function receives the segment matrix as argument and returns a likelihood estimation. This function is used to calculate the change points that maximize the total likelihood. Depending on the algorithm being used, this function is likely to be executed many times, in which case it's also likely to be the bottleneck of the function execution, so it's advised that this function should have fast implementation.

`max_segments` an integer that defines the maximum amount of segments to split the data into.

`allow_parallel` allows parallel execution to take place using the registered cluster. Assumes a cluster is registered with the `foreach` package. Defaults to TRUE.

Details

Function that implements the dynamic programming algorithm, with the intent of finding points of independent change points for which the cost function is minimized. It analyzes all possible combinations, returning the change points that are guaranteed to segment the data matrix in the change points minimize total cost. Because it analyzes all possible combinations of change points, it has a O-squared algorithm complexity, meaning it works in an acceptable computation time for small datasets, but it takes quite longer for datasets with many columns. For big datasets, `hieralg()` might be more adequate.

Value

a list of type `segmentr`, which has the two attributes:

- `changepoints`: a vector with the first index of each identified change point
- `segments`: a list of vectors, in which each vector corresponds to the indices that identifies a segment.

hieralg *Segment data into change points assuming hierarchical structure*

Description

By assuming change points follow an hierarchical architecture, this architecture manages to run faster by not searching all possible branches

Usage

```
hieralg(data, cost, likelihood, max_segments = ncol(data),
  allow_parallel = TRUE)
```

Arguments

`data`	matrix for which to find the change points
`cost`	a function receives the segment matrix as argument and returns a cost for the segment. This function is used to calculate the change points that minimize the total cost. Depending on the algorithm being used, this function is likely to be executed many times, in which case it's also likely to be the bottleneck of the function execution, so it's good for this function to have a fast implementation.
`likelihood`	deprecated: use cost instead. function receives the segment matrix as argument and returns a likelihood estimation. This function is used to calculate the change points that maximize the total likelihood. Depending on the algorithm being used, this function is likely to be executed many times, in which case it's also likely to be the bottleneck of the function execution, so it's advised that this function should have fast implementation.
`max_segments`	an integer that defines the maximum amount of segments to split the data into.
`allow_parallel`	allows parallel execution to take place using the registered cluster. Assumes a cluster is registered with the `foreach` package. Defaults to TRUE.

Details

Fast algorithm that segments data into change points, and it does so by simplifying by reducing the search possibilities by assuming data split in an hierarchical structure, i.e. a segment found in a first trial is assumed to contain only segments independent of the rest of the data. This algorithm usually runs very fast, but is known to yield less accurate results, possibly not finding the exact change points that would minimize cost.

Value

a list of type `segmentr`, which has the two attributes:

- `changepoints`: a vector with the first index of each identified change point
- `segments`: a list of vectors, in which each vector corresponds to the indices that identifies a segment.

hybridalg — *Segment data into change points using a mixed hierarchical-exact approach*

Description

For the larger datasets, assume the data is hierarchical, but calculate the exact segments when they're smaller than a threshold

Usage

```
hybridalg(data, cost, likelihood, allow_parallel = TRUE,
  max_segments = ncol(data), threshold = 50)
```

Arguments

`data` matrix for which to find the change points

`cost` a function receives the segment matrix as argument and returns a cost for the segment. This function is used to calculate the change points that minimize the total cost. Depending on the algorithm being used, this function is likely to be executed many times, in which case it's also likely to be the bottleneck of the function execution, so it's good for this function to have a fast implementation.

`likelihood` deprecated: use cost instead. function receives the segment matrix as argument and returns a likelihood estimation. This function is used to calculate the change points that maximize the total likelihood. Depending on the algorithm being used, this function is likely to be executed many times, in which case it's also likely to be the bottleneck of the function execution, so it's advised that this function should have fast implementation.

`allow_parallel` allows parallel execution to take place using the registered cluster. Assumes a cluster is registered with the `foreach` package. Defaults to TRUE.

`max_segments` an integer that defines the maximum amount of segments to split the data into.

`threshold` the threshold for which the exact algorithm will be used, i.e. when the number of columns in the segment is less than or equal to the threshold.

Details

This algorithm implements an approach mixing the hierarchical and exact algorithms. It uses the hierarchical algorithms when the size of the segment is bigger than the threshold, and then goes on to use the exact algorithm when the size of the segment is less than or equal to the threshold.

Value

a list of type `segmentr`, which has the two attributes:

- `changepoints`: a vector with the first index of each identified change point
- `segments`: a list of vectors, in which each vector corresponds to the indices that identifies a segment.

multivariate *Efficient Logarithmic Discrete Multivariate Likelihood estimation*

Description

Estimate the likelihood of a given segment using the discrete multivariate estimation, implemented efficiently in C++

Usage

```
multivariate(data, na_action = function(d) d[, colSums(is.na(d)) == 0,
  drop = FALSE])
```

Arguments

`data`	Matrix to estimate the multivariate of. Each row is considered to be an observation, and each column is considered to be a different variable.
`na_action`	A function that is applied to the `data` parameter. Defaults to removing columns with NA.

Details

Calculates the discrete log likelihood multivariate estimation of a data matrix using an algorithm implemented in C++ for performance. This is intended to be used in conjunction with `segment()`, as the log likelihood function is executed multiple times, which makes it the bottleneck of the computation. Because the multivariate is so commonly used, this efficient implementation is provided.

Value

the estimate of the Discrete Maximum Likelihood for the dataframe provided.

print.segmentr *Print a segmentr object*

Description

Prints a short description of the segments found in the `segmentr` object

Usage

```
## S3 method for class 'segmentr'
print(x, ...)
```

Arguments

x	an object of type segmentr, containing change point information
...	further arguments to be passed down to other methods

Details

A short representation of the segments is printed on the screen, using the `start:end` range notation.

Examples

```
make_segment <- function(n, p) matrix(rbinom(100 * n, 1, p), nrow = 100)
data <- cbind(make_segment(5, 0.1), make_segment(10, 0.9), make_segment(2, 0.1))
heterogeneity_cost <- function(X) sum((X - mean(X))^2) + 1
x <- segment(data, cost = heterogeneity_cost, algorithm = "hieralg")
print(x)
```

r_multivariate *Logarithmic Discrete Multivariate Likelihood estimation function implemented in R*

Description

Estimate the likelihood of a given segment using the discrete multivariate estimation, but code runs more slowly due to R implementation

Usage

```
r_multivariate(data, na.omit = TRUE)
```

Arguments

data	Matrix to estimate the multivariate of. Each row is considered to be an observation, and each column is considered to be a different variable.
na.omit	If true, omits NAs from the dataset.

Details

This log likelihood function is implemented in R in order to be used to benchmark against the `multivariate()` version implemented in C++ for performance.

Value

The estimate of the Discrete Maximum Likelihood for the dataframe provided.

`segment` *Segment data into change points*

Description

Generic function to segment data into separate change points according to specified algorithm

Usage

```
segment(data, cost, likelihood, max_segments = ncol(data),
  allow_parallel = TRUE, algorithm = "exact", ...)
```

Arguments

`data`	matrix for which to find the change points
`cost`	a function receives the segment matrix as argument and returns a cost for the segment. This function is used to calculate the change points that minimize the total cost. Depending on the algorithm being used, this function is likely to be executed many times, in which case it's also likely to be the bottleneck of the function execution, so it's good for this function to have a fast implementation.
`likelihood`	deprecated: use cost instead. function receives the segment matrix as argument and returns a likelihood estimation. This function is used to calculate the change points that maximize the total likelihood. Depending on the algorithm being used, this function is likely to be executed many times, in which case it's also likely to be the bottleneck of the function execution, so it's advised that this function should have fast implementation.
`max_segments`	an integer that defines the maximum amount of segments to split the data into.
`allow_parallel`	allows parallel execution to take place using the registered cluster. Assumes a cluster is registered with the `foreach` package. Defaults to TRUE.
`algorithm`	can be of type `exact`, `hierarchical` or `hybrid`, Default: `exact`
`...`	other parameters to be passed to the underlying function

Details

This function can be used as a generic function to call any of the algorithms implemented by the package. Depending on the type of data the user wants to segment, one algorithm might be more adequate than the others.

Value

a list of type `segmentr`, which has the two attributes:

- `changepoints`: a vector with the first index of each identified change point
- `segments`: a list of vectors, in which each vector corresponds to the indices that identifies a segment.

See Also

exactalg() for the exact algorithm, hieralg() for the hierarchical algorithm implementation, hybridalg() for the hybrid algorithm implementation.

Examples

```
make_segment <- function(n, p) matrix(rbinom(100 * n, 1, p), nrow = 100)
data <- cbind(make_segment(5, 0.1), make_segment(10, 0.9), make_segment(2, 0.1))
heterogeneity_cost <- function(X) sum((X - mean(X))^2) + 1
segment(data, cost = heterogeneity_cost, algorithm = "hieralg")
```

Index