Package ‘dpseg’

October 13, 2022

Title  Piecewise Linear Segmentation by Dynamic Programming
Version  0.1.1
Maintainer  Rainer Machne <machne@hhu.de>

BugReports  https://gitlab.com/raim/dpseg/-/issues

Description  Piecewise linear segmentation of ordered data by a
dynamic programming algorithm. The algorithm was developed for time
series data, e.g. growth curves, and for genome-wide read-count data
from next generation sequencing, but is broadly applicable.
Generic implementations of dynamic programming routines allow
to scan for optimal segmentation parameters and test custom
segmentation criteria ("scoring functions").

Depends  R (>= 3.0.0)
Imports  Rcpp (>= 0.12.18)
Suggests  markdown, knitr, htmltools, RcppDynProg, microbenchmark,
ggplot2

LinkingTo  Rcpp

URL  https://gitlab.com/raim/dpseg/
License  GPL (>= 2)
Encoding  UTF-8
LazyData  true
VignetteBuilder  knitr

RoxygenNote  6.1.1
NeedsCompilation  yes

Author  Rainer Machne [aut, cre] (<https://orcid.org/0000-0002-1274-5099>),
        Peter F. Stadler [aut]

Repository  CRAN
Date/Publication  2020-08-17 13:20:02 UTC
R topics documented:

- addLm .......................................................... 2
- backtrace_r ..................................................... 3
- dpseg ............................................................. 4
- dpseg_old ....................................................... 6
- estimateP ....................................................... 8
- movie ............................................................. 8
- oddata ........................................................... 10
- plot.dpseg ...................................................... 10
- predict.dpseg .................................................. 11
- print.dpseg ..................................................... 11
- scanP ............................................................ 12
- sgtable .......................................................... 13

Index 14

addLm

Add linear regression data to dpseg results or a table of segment borders.

Description

addLm takes a segment table (with start/end columns) or a result object from code dpseg, calls base R function lm for each segment, and adds slope, intercept, r2 and variance of residuals to the segment table. This data is required for plot and predict method, eg. when dpseg was called with a pre-calculated scoring matrix, or alternative scoring functions or recursion.

Usage

addLm(dpseg, x, y)

Arguments

dpseg result object (class "dpseg") returned by function dpseg or simply a segment table with "start" and "end" indices

x original x-data used

y original y-data used

Value

Returns the input dpseg object or segment table, but with original xy data and fit results from a linear regression with base R (lm(y~x)) added to the results and linear regression coefficient and goodness of fit meaurs in the main segments table.
Examples

```r
## 1: run dpseg with store.matrix=TRUE to allow re-rung
segs <- dpseg(x=oddata$Time, y=log(oddata$A3), store.matrix=TRUE)

## 2: run dpseg with score function matrix input
segr <- dpseg(y=segs$SCR, P=0.0001, verb=1)

## NOTE: only data indices i and j are provided in results
print(segr)

## 3: add original data and linear regression for segments
## NOTE: now also plot and predict methods work
segr <- addLm(segr, x=oddata$Time, y=log(oddata$A3))
print(segr)
```

Description

Backtracing segment borders from the `imax` vector of a `dpseg` recursion. This function is implemented more efficiently in Rcpp; the R code is kept for documentation, benchmarking and development.

Usage

```r
backtrace_r(imax, jumps = 0)
```

Arguments

- `imax`: integer vector of segment borders as returned by `dpseg` recursion functions
- `jumps`: allows discontinuous jumps: move 1 index position back, only for $S_{i-1} + score(i,j)$

Value

an integer vector of segment ends
dpseg

*dpseg* : linear segmentation by dynamic programming

**Description**

dpseg splits a curve (x,y data) into linear segments by a straightforward dynamic programming recursion:

\[ S_j = \max(S_{i-\text{jumps}} + \text{score}(i,j) - P) \]

where score is a measure of the goodness of the fit of a linear regression (equiv. to \(\text{lm}(y\sim x)\)) between data points \(i < j\). The default scoring function is simply the negative variance of residuals of the linear regression (see arguments *type* and *scoref*). \(P\) is a break-point penalty that implicitly regulates the number of segments (higher \(P\): longer segments), and jumps==1 allows for disjoint segments. The arguments *minl* and *maxl* specify minimal \((i \leq j - \text{minl})\) and maximal \((i \geq j - \text{maxl})\) segment lengths, which allows to significantly decrease memory usage when expected segment lengths are known.

**Usage**

dpseg(x, y, maxl, jumps = FALSE, P = 0, minl = 3, S0 = 1, type = "var", scoref, verb = 1, move, store.values = TRUE, store.matrix = FALSE, add.lm = FALSE, recursion, backtrace, ...)

**Arguments**

- **x** : x-values, not used if *y* is a scoring function matrix
- **y** : y-values, or a pre-calculated scoring function matrix \(SCR_{i,j}\) (eg. from a previous run of dpseg). See section "Value" below for details on the structure \(SCR_{i,j}\).
- **maxl** : maximal segment length, \(i \geq j - \text{maxl}\)
- **jumps** : allow for jumps between segments, if TRUE segment ends are 1 index left of the segment starts
- **P** : break-point penalty, increase to get longer segments with lower scores (eg. higher residual variance)
- **minl** : minimal segment length, \(i \leq j - \text{minl}\)
- **S0** : initialization of \(S_0\), choose high enough to avoid length 1 cutoffs at start
- **type** : type of scoring function: available are "var" for "variance of residuals", "cor" for Pearson correlation, or "r2" for r-squared; see the package vignette("dpseg") for details.
- **scoref** : alternative scoring function
- **verb** : print progress messages
- **move** : logical indicating whether move is required in backtracing, required for the alternative recursion \(S_i + \text{score}(i+1,j)\)
- **store.values** : store scoring values (linear regression results)
store.matrix  store the fitscore matrix
add.lm  add a linear fit using R base lm for final segments; may save memory/speed if store.values==FALSE
recursion  internal recursion function to be used for segmentation; used for debugging, benchmarking and development, and required for putative novel scoring functions scoref
backtrace  internal function to be used for back-tracing; used for debugging, benchmarking and development, and may be required to test novel scoring functions scoref and/or recursion
... further arguments to recursion

Details

See the vignette("dpseg") for the theory and details on the choice of scoring functions and selection of the penalty parameter P.

Value

Returns a list object of class dpseg (with print.dpseg plot.dpseg and predict.dpseg methods). The main result of the algorithm is a table (data.frame) of predicted segments in list object segments. The original data, run parameters and (optionally) additional data calculated and used by the algorithm are also returned.

segments: main result table: a data.frame that lists the start and end x-values of the segments, the start and end indices (i,j) in the data vectors, the linear regression coefficients and goodness-of-fit measures for the segments (intercept, slope, r-squared, variance of residuals). If dpseg was called with a pre-calculated scoring matrix, the table only contains start and end indices i,j. If option add.lm=TRUE or the result object was sent through function addLm the table additionally contains results from R’s lm, indicated by an ".lm" suffix.

S: results of the recursion, ie. S_j in above equation.

imax: vector j=1,…,n, storing the i_{max} that yielded S_j, ie., the sole input for the backtracing function.

values: linear regression coefficients and measures for the segment ending at j and starting at i_{max}(j). Only present if store.values=TRUE.

SCR: scoring function matrix SCR_{i,j} = score(i,j) where positions j are the columns and i the rows; a banded matrix with non-NA values between i ≤ j - minl and i ≥ j - maxl. Note, that this matrix can be re-used in subsequent calls as dpseg(y=previous$SCR) which runs much faster and allows to efficiently scan for alternative parameters. Only present if store.matrix=TRUE.

fits: result objects from lm. Only present if add.lm=TRUE.

traceback: result of the call to the backtracing function: ends of the segments.

xy: original x/y data (xy.coords).

removed: index of NA/Inf values that were removed before running the algorithm.

parameters: used parameters P, jumps, maxl and minl.
Dependencies

The package strictly depends only on RcppEigen. All other dependencies are usually present in a basic installation (stats, graphics, grDevices).

Author(s)

Rainer Machne <machne@hhu.de>, Peter F. Studler <studla@bioinf.uni-leipzig.de>

Examples

```r
## calculate linear segments in semi-log bacterial growth data
## NOTE: library loads bacterial growth curve data as data.frame oddata
segs <- dpseg(x=oddata$Time, y=log(oddata$A3), minl=5, P=0.0001, verb=1)

## inspect resulting segments
print(segs)

## plot results (also see the movie method)
plot(segs, delog=TRUE, log="y")

## predict method
plot(predict(segs), type="l")
```

Description

See dpseg for a current version of this algorithm. Note: this was a first test implementation of the linear piecewise segmentation by a dynamic programming approach. This implementation is very slow. A much more efficient version, dpseg, calculates the variance of residuals of a linear regression incrementally while looping through the recursion, and is implemented in Rcpp. See there for details on the algorithm. This version is kept alive, since it is a more general implementation, allowing to test different regression and scoring functions by command-line arguments.

Usage

```r
dpseg_old(x, y, minl, maxl = length(x), P = 0, EPS,
             store.matrix = FALSE, fitscoref = fitscore, fitf = linregf,
             scoref = varscore, verb = 0)
```

Arguments

- `x`: x-values
- `y`: y-values
- `minl`: minimal segment length

---

**dpseg_old**

inefficient dpseg implementation
**dpseg_old**

maxl
maximal segment length

P
jump penalty, increase to get fewer segments # @inheritParams score

EPS
a pre-calculated fitscore matrix, will be generated if missing

store.matrix
store the fitscore matrix

fitscoref
the heavy-load loop that fills the fitscore matrix using fitf and scoref

fitf
fit function, used in the scoring function scoref: (TODO: currently expecting a fit object that provides intercept and slope as coef(obj)[1:2] only for the result table)

scoref
function to calculate a score from the passed fit function

verb
print progress messages

Details

The recursion calculates $S_j = \max(S_i + \text{fitscore}(i + 1, j)) - P$, where the fitscore is the variance of the residuals of a linear regression ($\text{lm}(y \sim x)$) between $x_{i+1}$ to $x_j$, $P$ is a jump penalty that implicitly regulates the number of segments, minl and maxl are minimal and maximal lengths of segments. Uses **RcppEigen::fastLm** for linear regression.

Value

Returns a list of result structures very similar to the list of class "dpseg" returned by function **dpseg**, except for the name of the scoring function matrix, here: EPS. See ?dpseg for detailed information on these structures.

Examples

```r
## NOTE: not run because it's too slow for R CMD check --as-cran
calculate linear segments in semi-log bacterial growth data
## NOTE: library loads bacterial growth curve data as data.frame oddata
Sj <- dpseg_old(x=oddata$Time, y=log(oddata$A3), minl=5, P=0.0001, verb=1)

## inspect resulting segments
print(Sj)

## plot results
plot(Sj, delog=TRUE, log="y")

## NOTE: predict method & movie function do not work for dpseg_old
```
### estimateP

Estimate a starting value for penalty \( P \).

**Description**

The break-point penalty \( P \) in a `dpseg` recursion, should be in the range of expected values of the scoring function. To find a good initial estimate for \( P \) when using the default scoring function (see `dpseg`), the data is smoothed by `smooth.spline` and the variance of residuals reported.

**Usage**

```r
estimateP(x, y, plot = FALSE, ...)
```

**Arguments**

- `x`: x-values
- `y`: y-values
- `plot`: plot the `smooth.spline`
- `...`: parameters for `smooth.spline`

**Value**

Returns a double, variance of residuals of a spline fit (\( \text{var}(\text{smooth.spline}(x, y, \ldots) \cdot y - y) \))

**Examples**

```r
x <- as.data.frame$Time
y <- log(as.data.frame$A5)
p <- estimateP(x=x, y=y, plot=TRUE)
plot(dpseg(x=x, y=y, jumps=TRUE, P=round(p,3)))
```

---

### movie

Visualizes the `dpseg` segmentation recursion as a movie.

**Description**

Generates a movie of the calculation steps \( j = 1, \ldots, n \) while looping through the recursion \( S_j \). Plots are sent to the active plot device or, if `path` is specified, to a video file `<path>/<file.name>.<format>` via a system call to Image Magick’s `convert`. Saving to a file likely only works on Linux systems with Image Magick installed and `convert` available in the `PATH` environment variable. `format` are formats available for `convert`, eg. `format="gif"` or `format="mpeg"`. See the vignette(“dpseg”) for details on the plotted data.
Usage

movie(dpseg, fix.ylim = TRUE, frames, delay = 0.1, repeat.last = 5,
      ylab = "scoring function", ylab2 = "y", xlab = "x", path,
      file.name = "dpseg_movie", format = "gif", res = 200, ...)  

Arguments

dpseg                         result object of class dpseg returned by function dpseg
fix.ylim                      fix the y-axis of the score function
frames                       x range to show as movie frames
delay                        delay between frames in seconds, between x11 plot updates or as argument
                              -delay to the system call to Image Magicks convert
repeat.last                  repeat list frame this many times
ylab                         left y-axis label, for the scoring function
ylab2                        right y-axis label, for the original data
xlab                         x-axis label
path                         path where both temporary jpeg files and the final movie file will be generated.
                              If not specified the indidvual frames will be plotted to the active plot device.
file.name                    name of the generated video file <path>/<file.name>.<format>
format                       format of the video, all outputs that image magick's convert can generate, e.g.
                              "mpg" or "gif"
res                          resolution of the generated movie (pixels per inch)
...                          arguments passed to default plot function

Examples

## NOTE: requires that dpseg is run with store.matrix=TRUE
segs <- dpseg(x=oddata$Time, y=log(oddata$A3), minl=5, P=0.0001, store.matrix=TRUE)

## View the algorithm in action:
movie(segs, delay=0)

## NOTE: if Image Magicks convert is installed you can set the path
## option to save the movie as <path>/<file.name>.<format>, where format
## can be "gif", "mpeg" or else, depending on the Image Magick installation.
oddata

*Escherichia coli growth curves.*

**Description**

Optical density (OD) data from a 96-well microtiter plate experiment, growing Escherichia coli cells in M9 medium in a BMG Optima platereader.

**Usage**

oddata

**Format**

A data frame with the measurement time in column 1 and bacterial growth data (or blanks) in 2:ncol(oddata). Column names correspond to the well on the microtiter plate.

**Source**

Tom Rohr, Anna Behle, Rainer Machne, HHU Duesseldorf, 2018

---

**plot.dpseg**

*Plot method for a dpseg segmentation model.*

**Description**

Plot method for a dpseg segmentation model.

**Usage**

```r
## S3 method for class 'dpseg'
plot(x, delog = FALSE, col, main, ...)
```

**Arguments**

- `x` result object returned by function `dpseg`
- `delog` plot exp(y)
- `col` optional color vector for segments
- `main` plot title, dpseg parameters will be plotted if missing
- `...` arguments passed to default `plot` function

**Value**

Silently returns the `x$segments` table, with color values added if they were missing in the input.
predict.dpseg

Predict method for ‘dpseg’ segmentations

Description

Predicted values based on a data segmentation model from dpseg.

Usage

```r
## S3 method for class 'dpseg'
predict(object, xout, ...)  
```

Arguments

- **object**: result object returned by function `dpseg`
- **xout**: new x-values at which to predict \( \hat{y} \)
- **...**: not used

Value

Returns predicted linear segments as x,y coordinates (grDevices::xy.coords) at xout.

Examples

```r
x <- oddata$Time
y <- log(oddata$A5)
segs <- dpseg(x=x, y=y, P=0.0001)
## predict method
plot(x=x, y=y, pch=19, cex=0.5)
lines(predict(segs), col=2, lwd=2)
```

print.dpseg

Print method for linear segmentation result from `dpseg`.

Description

Prints the main result table x$segments, segment coordinates and indices, and parameters from the recursion. See `dpseg` for details.

Usage

```r
## S3 method for class 'dpseg'
print(x, ...)
```
Arguments

x result object returned by function dpseg
...

further arguments to print.data.frame

scanP

Scan over different penalty $P$ values

Description

Runs the dpseg recursion for different values of the penalty parameter $P$ and returns a matrix with the used $P$ values, the resulting number of segments and (optionally) the median of segment variance of residuals.

Usage

scanP(x, y, P, var = TRUE, use.matrix = TRUE, plot = TRUE, verb = 1, ...)

Arguments

x x-values
y y-values
P vector of penalties $P$ to scan
var add the median of the variances of residuals of all segments to output (save time by var=FALSE)
use.matrix use the stored scoring function matrix for more efficient scans; set this to FALSE if you run into memory problems
plot plot results
verb print progress messages
...
parameters for dpseg (except P)

Value

Returns a matrix with the penalties $P$ in the first column, the number of segments in the second column and the median of variances in the third column.

Examples

x <- oddata$Time
y <- log(oddata$A5)
par(mai=c(par("mai")[1:3], par("mai")[2])) # to show right axis
sp <- scanP(x=x, y=y, P=seq(-.01,.1,length.out=50), plot=TRUE)
sgtable

Construct a segment table

Description

Constructs a segment table from segment ends (imax) returned by dpseg backtracing functions backtrace_r and backtrace_c. Correct segment break-points require to know whether segment recursion was run with the jumps option of dpseg. In joint segments (jumps=FALSE) segment borders are part of both left and right segments.

Usage

sgtable(ends, starts, jumps = TRUE)

Arguments

ends integer vector of segment ends
starts integer vector of segment starts
jumps same parameter as passed to recursion function, allowing for discontinous jumps (TRUE) or enforcing joint segments (FALSE)

Value

a table with segment start and end columns
Index

* datasets
  - oddata, 10

addLm, 2, 5

backtrace_r, 3, 13

dpseg, 2, 3, 4, 6–13
dpseg-package (dpseg), 4
dpseg_old, 6

estimateP, 8

lm, 2

movie, 8

oddata, 10

plot, 9, 10
plot.dpseg, 10
predict.dpseg, 11
print.dpseg, 11

RcppEigen:fastLm, 7

scanP, 12
sgtable, 13
smooth.spline, 8