Package 'cpop'

August 23, 2022

Type Package

Title Detection of Multiple Changes in Slope in Univariate Time-Series

Version 1.0.3

Date 2022-08-23

Maintainer Daniel Grose <dan.grose@lancaster.ac.uk>

Description Detects multiple changes in slope using the CPOP dynamic programming approach of Fearnhead, Maidstone, and Letchford (2018) <doi:10.1080/10618600.2018.1512868>. This method finds the best continuous piecewise linear fit to data under a criterion that measures fit to data using the residual sum of squares, but penalizes complexity based on an L0 penalty on changes in slope.

License GPL (>= 2)

Depends R (>= 3.5.0),crops,pacman

Imports Rdpack,methods, Rcpp (>= 0.12.13), ggplot2,mathjaxr,pracma

LinkingTo Rcpp

Suggests testthat,R.rsp

RoxygenNote 7.1.2

RdMacros Rdpack,mathjaxr

VignetteBuilder R.rsp

R topics documented:

`changepoints` *Changepoint Locations*

Description

Creates a data frame containing the locations of the changepoints in terms of the index of the data and the value of the location at that index.

Arguments

`object`	An instance of an cpop S4 class produced by cpop.

Value

A data frame.

Examples

```
library(cpop)

# generate some test data
set.seed(0)
x <- seq(0,1,0.01)
n <- length(x)
sd <- rep(0.1,n)
mu <- c(2*x[1:floor(n/2)],2 - 2*x[(floor(n/2)+1):n])
y <- rnorm(n,mu,sd)

# use the locations in x
res <- cpop(y,x,beta=2*log(length(y)),sd=sd)
changepoints(res)
```

`cost` *A function for calculating the cost of a model fitted by cpop*

Description

Calculates the penalised cost of a model fitted by cpop using the residual sum of squares and the penalty values.

Usage

```
cost(object)
```

Arguments

object	An instance of an S4 class produced by cpop.

Value

Numerical value of the penalised cost associated with the segmentations determined by using cpop

Examples

```
library(cpop)

# simulate data with change in gradient
set.seed(1)
x <- (1:50/5)^2
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=1)

# determine changepoints
res <- cpop(y,x,beta=2*log(length(y)))

# calculate the penalised cost
cost(res)
```

---

cpop *cpop*

Description

The CPOP algorithm fits a change-in-slope model to data. It assumes that we have we have data points $(y_1, x_1), \ldots, (y_n, x_n)$, ordered so that $x_1 < x_2 < \cdots < x_n$. For example x_i could be a time-stamp of when response y_i is obtained. We model the response, y, as a signal plus noise where the signal is modelled as a continuous piecewise linear function of x. That is

$$y_i = f(x_i) + \epsilon_i$$

where $f(x)$ is a continuous piecewise linear function.

To estimate the function $f(x)$ we specify a set of N grid points, $g_{1:N}$ with these ordered so that $g_i < g_j$ if and only if $i < j$, and allow the slope of $f(x)$ to only change at these grid points. We then estimate the number of changes, the location of the changes, and hence the resulting function $f(x)$ by minimising a penalised weighted least squares criteria. This criteria includes a term which measures fit to the data, which is calculated as the sum of the square residuals of the fitted function, scaled by the variance of the noise. The penalty is proportional to the number of changes.

That is our estimated function will depend on K, the number of changes in slope, their locations, $\tau_1, \ldots, \tau_K$, and the value of the function $f(x)$ at these change points, $\alpha_1, \ldots, \alpha_K$, and its values,

α_0 at $\tau_0 < x_1$ and α_{K+1} at some $\tau_{K+1} > x_N$. The CPOP algorithm then estimates K, $\tau_{1:K}$ and $\alpha_{0:K+1}$ as the values that solve the following minimisation problem

$$\min_{K,\tau_{1:K}\in g_{1:N},\alpha_{0:K+1}} \left\{ \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \left(y_i - \alpha_{j(i)} - (\alpha_{j(i)+1} - \alpha_{j(i)}) \frac{x_i - \tau_{j(i)}}{\tau_{j(i)+1} - \tau_{j(i)}} \right)^2 + K\beta \right\}$$

where $\sigma_1^2, \ldots, \sigma_n^2$ are the variances of the noise ϵ_i for $i = 1, \ldots, n$, and β is the penalty for adding a changepoint. The sum in this expression is the weighted residual sum of squares, and the $K\beta$ term is the penalty for having K changes.

If we know, or have a good estimate of, the residual variances, and the noise is (close to) independent over time then an appropriate choice for the penalty is $\beta = 2\log n$, and this is the default for CPOP. However in many applications these assumptions will not hold and it is advised to look at segmentations for different value of β – this is possible using CPOP with the CROPS algorithm `cpop.crops`. Larger values of β will lead to functions with fewer changes. Also there is a trade-off between the variances of the residuals and β: e.g. if we double the variances and half the value of β we will obtain the same estimates for the number and location of the changes and the underlying function.

Usage

```
cpop(
  y,
  x = 1:length(y) - 1,
  grid = x,
  beta = 2 * log(length(y)),
  sd = 1,
  minseglen = 0,
  prune.approx = FALSE
)
```

Arguments

`y`	A vector of length n containing the data.
`x`	A vector of length n containing the locations of y. Default value is NULL, in which case the locations `x = 1:length(y)-1` are assumed.
`grid`	An ordered vector of possible locations for the change points. If this is NULL, then this is set to x, the vector of times/locations of the data points.
`beta`	A positive real value for the penalty incurred for adding a changepoint (prevents over-fitting). Default value is `2*log(length(y))`.
`sd`	Estimate of residual standard deviation. Can be a single numerical value or a vector of values for the case of varying standard deviation. Default value is 1.
`minseglen`	The minimum allowable segment length, i.e. distance between successive changepoints. Default is 0.
`prune.approx`	Only relevant if a minimum segment length is set. If True, cpop will use an approximate pruning algorithm that will speed up computation but may occasionally lead to a sub-optimal solution in terms of the estimate change point locations. If the minimum segment length is 0, then an exact pruning algorithm is possible and is used.

Details

Algorithm for finding the best segmentation of data for a change-in-slope model.

Value

An instance of an S4 class of type cpop.class.

References

Fearnhead P, Maidstone R, Letchford A (2019). "Detecting Changes in Slope With an L0 Penalty." *Journal of Computational and Graphical Statistics*, **28**(2), 265-275. doi:10.1080/10618600.2018.1512868.

Examples

```
library(cpop)

# simulate data with change in gradient
set.seed(1)
x <- (1:50/5)^2
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=1)
# analyse using cpop
res <- cpop(y,x)
p <- plot(res)
print(p)

# generate the "true" mean
mu <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=0)
# add the true mean to the plot
library(pacman)
p_load(ggplot2)
p <- p + geom_line(aes(y = mu), color = "blue", linetype = "dotted")
print(p)

# heterogeneous data
set.seed(1)
sd <- (1:50)/25
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=sd)

# analysis assuming constant noise standard deviation
res <- cpop(y,x,beta=2*log(length(y)),sd=sqrt(mean(sd^2)))
p <- plot(res)
print(p)

# analysis with the true noise standard deviation
res.true <- cpop(y,x,beta=2*log(length(y)),sd=sd)
p <- plot(res.true)
print(p)

# add the true mean to the plot
p <- p + geom_line(aes(y = mu), color = "blue", linetype = "dotted")
print(p)
```

cpop.crops *Calculate changepoint locations over a range of penalty values*

Description

Runs the Changepoints for a Range of Penalties (CROPS) algorithm of Haynes et al. (2017) to find all of the optimal segmentations for multiple penalty values over a continuous range. For details of the CROPS method see the **crops** package. To obtain details for each segmentation determined by `cpop.crops` use the `segmentations` method (see example below).

Usage

```
cpop.crops(
  y,
  x = 1:length(y),
  grid = x,
  beta_min = 1.5 * log(length(y)),
  beta_max = 2.5 * log(length(y)),
  sd = 1,
  minseglen = 0,
  prune.approx = FALSE
)
```

Arguments

`y`	A vector of length n containing the data.
`x`	A vector of length n containing the locations of y. Default value is NULL, in which case the locations `x = 1:length(y)-1` are assumed.
`grid`	An ordered vector of possible locations for the change points. If this is NULL, then this is set to x, the vector of times/locations of the data points.
`beta_min`	Minimum value of the penalty range to be searched. Default is `1.5*log(length(y))`
`beta_max`	Maximum value of the penalty range to be searched. Default is `2.5*log(length(y))`
`sd`	Estimate of residual standard deviation. Can be a single numerical value or a vector of values for the case of varying standard deviation. Default value is 1.
`minseglen`	The minimum allowable segment length, i.e. distance between successive changepoints. Default is 0.
`prune.approx`	Only relevant if a minimum segment length is set. If TRUE, cpop will use an approximate pruning algorithm that will speed up computation but may occasionally lead to a sub-optimal solution in terms of the estimate change point locations. If the minimum segment length is 0, then an exact pruning algorithm is possible and is used.

Value

An instance of an S4 class of type cpop.crops.class which extends the crops.class in **crops**.

References

Haynes K, Eckley IA, Fearnhead P (2017). "Computationally Efficient Changepoint Detection for a Range of Penalties." *Journal of Computational and Graphical Statistics*, **26**(1), 134-143. doi:10.1080/10618600.2015.1116445.

Grose D, Fearnhead P (2021). *crops: Changepoints for a Range of Penalties (CROPS)*. R package version 1.0.1, https://CRAN.R-project.org/package=crops.

Examples

```
library(cpop)

# generate some data
set.seed(0)
x <- seq(0,1,0.01)
n <- length(x)
sd <- rep(0.1,n)
mu <- c(2*x[1:floor(n/2)],2 - 2*x[(floor(n/2)+1):n])
y <- rnorm(n,mu,sd)
# calculate the changepoint locations and cost over a range of penalty values
res <- cpop.crops(y,x,sd=sd,beta_min=0.4*log(length(y)),beta_max=2.5*log(length(y)))
summary(res)

# plot the results
plot(res)

# show the segmentations
 segmentations(res)
```

`cpop.crops.models` *Obtain the cpop models created by cpop.crops*

Description

Obtains a list of models corresponding to each of the beta (penalty) values considered during a `cpop.crops` analysis.

Usage

```
cpop.crops.models(object)
```

Arguments

`object`	An S4 object of type `cpop.crops.class` produced by `cpop.crops`.

Value

A list of S4 cpop.class objects corresponding to the beta values considered by a cpop.crops analysis.

References

Haynes K, Eckley IA, Fearnhead P (2017). "Computationally Efficient Changepoint Detection for a Range of Penalties." *Journal of Computational and Graphical Statistics*, **26**(1), 134-143. doi:10.1080/10618600.2015.1116445.

Grose D, Fearnhead P (2021). *crops: Changepoints for a Range of Penalties (CROPS)*. R package version 1.0.1, https://CRAN.R-project.org/package=crops.

See Also

cpop.crops,crops

Examples

```
library(cpop)

set.seed(1)
n <- 500
x <- 1:n
m <- 10
mu <- simchangeslope(x,changepoints=(n/(m+1))*0:m,change.slope=c(0.1,0.2*(-1)^(1:m)),sd=0)
epsilon <- rnorm(n+2)
y <- mu+(epsilon[1:n]+epsilon[2:(n+1)]+epsilon[3:(n+2)])/sqrt(3)
res.crops <- cpop.crops(y,x,beta_min=0.5*log(length(y)),beta_max=40*log(length(y)))
models <- cpop.crops.models(res.crops)
for(m in models)
{
  plot(m)
}
```

---

estimate *A function for estimating the fit of a cpop model*

Description

Estimates the fit of a cpop model at the specified locations. If no locations are specified it evaluates the estimates at the locations specified when calling cpop.

Usage

```
estimate(object, x = object@x, ...)
```

Arguments

object	An instance of an S4 class produced by cpop.
x	Locations at which the fit is to be estimated. Default value is the x locations at which the cpop object was defined.
...	Additional arguments.

Value

A data frame with two columns containing the locations x and the corresponding estimates y_hat.

Examples

```
library(cpop)

# simulate data with change in gradient
set.seed(1)
x <- (1:50/5)^2
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=1)

# determine changepoints
res <- cpop(y,x,beta=2*log(length(y)))

# estimate fit at points used in call to cpop
estimate(res)

# estimate fit at specified locations
estimate(res,seq(0,100,10))

#extrapolate fit
estimate(res,seq(-20,140,20))
```

---

fitted *Extract Model Fitted Values*

Description

Extracts the fitted values produced by cpop.

Usage

```
fitted(object)
```

Arguments

object	An instance of an S4 class produced by cpop.

Value

A data frame containing the endpoint coordinates for each line segment fitted between the detected changepoints. The data frame also contains the gradient and intercept values for each segment and the corresponding residual sum of squares (RSS).

Examples

```
library(cpop)

# simulate data with change in gradient
set.seed(1)
x <- (1:50/5)^2
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=1)

# determine changepoints
res <- cpop(y,x,beta=2*log(length(y)))

# calculate segments
fitted(res)
```

---

plot *Visualisation of Changepoint Locations and Data*

Description

Plot methods for S4 objects returned by cpop.

Usage

```
## S4 method for signature 'cpop.class,ANY'
plot(x)
```

Arguments

x	An instance of an cpop S4 class produced by cpop.

Value

A ggplot object.

Examples

```
library(cpop)

# simulate data with change in gradient
set.seed(1)
x <- (1:50/5)^2
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=1)
```

```
# analyse data
res <- cpop(y,x,beta=2*log(length(y)))

# generate plot object
p <- plot(res)

# visualise
print(p)
```

`residuals` *Extracts residuals from a cpop model*

Description

Extracts residuals from a cpop model.

Usage

```
## S4 method for signature 'cpop.class'
residuals(object)
```

Arguments

`object`	An instance of an S4 class produced by cpop.

Value

A single column matrix containing the residuals.

Examples

```
library(cpop)

# simulate data with change in gradient
set.seed(1)
x <- (1:50/5)^2
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=1)

# determine changepoints
res <- cpop(y,x,beta=2*log(length(y)))

# calculate the residuals
residuals(res)
```

show *Displays an S4 object produced by cpop.*

Description

Displays an S4 object produced by cpop.

Usage

```
show(object)
```

Arguments

object	An instance of an S4 class produced by cpop.

Examples

```
library(cpop)

# simulate data with change in gradient
set.seed(1)
x <- (1:50/5)^2
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=1)

# determine changepoints
res <- cpop(y,x,beta=2*log(length(y)))

# display a summary of the results using show
show(res)
```

simchangeslope *A function for generating simulated data*

Description

Generates simulated data for use with cpop.

Usage

```
simchangeslope(x, changepoints, change.slope, sd = 1)
```

Arguments

x	A numeric vector containing the locations of the data.
changepoints	A numeric vector of changepoint locations.
change.slope	A numeric vector indicating the change in slope at each changepoint. The initial slope is assumed to be 0.
sd	The residual standard deviation. Can be a single numerical value or a vector of values for the case of varying residual standard deviation. Default value is 1.

Value

A vector of simulated y values.

Examples

```
library(cpop)
library(pacman)
p_load(ggplot2)

# simulate changepoints with constant sd
set.seed(1)
changepoints <- c(0,25,50,100)
change.slope <- c(0.2,-0.3,0.2,-0.1)
x <- 1:200
sd <- 0.2
y <- simchangeslope(x,changepoints,change.slope,sd)
df <- data.frame("x"=x,"y"=y)
p <- ggplot(data=df,aes(x=x,y=y))
p <- p + geom_point()
p <- p + geom_vline(xintercept=changepoints,
                    color="red",
                    linetype="dashed")
print(p)

# simulate changepoints with varying sd
sd <- 0.2 + x/200
y <- simchangeslope(x,changepoints,change.slope,sd)
df$y <- y
p <- ggplot(data=df,aes(x=x,y=y))
p <- p + geom_point()
p <- p + geom_vline(xintercept=changepoints,
                    color="red",
                    linetype="dashed")
print(p)
```

summary *Summary of cpop Analysis.*

Description

Summary method for results produced by cpop.

Usage

```
## S4 method for signature 'cpop.class'
summary(object)
```

Arguments

object	An instance of an S4 class produced by cpop.

Examples

```
library(cpop)

# simulate data with change in gradient
set.seed(1)
x <- (1:50/5)^2
y <- simchangeslope(x,changepoints=c(10,50),change.slope=c(0.25,-0.25),sd=1)

# determine changepoints
res <- cpop(y,x,beta=2*log(length(y)))

# display a summary of the results
summary(res)
```

wavenumber_spectra *Wavenumber Power Spectra data.*

Description

Data of power spectra of velocity as a function of wavenumber obtained from climate models of the Atlantic Ocean for two different months and two climate scenarios: a present-day one (run 2000) and a future scenario (run 2100). See Richards KJ, Whitt DB, Brett G, Bryan FO, Feloy K, Long MC (2021). "The Impact of Climate Change on Ocean Submesoscale Activity." *Journal of Geophysical Research: Oceans*, **126**(5). doi:10.1002/essoar.10503524.1.

Usage

```
data(wavenumber_spectra)
```

Format

A dataframe with 5 columns and 247 rows. The first column is wavenumber (cycles per metre). The other columns are the power spectra values for different months (Feb run 2000, Aug run 2000, Feb run 2100, Aug run 2100). The original data is documented in Richards KJ, Whitt DB, Brett GJ (2020). "Climate change impact on submesoscale ROMS data." doi:10.5281/zenodo.4615129.

References

Richards KJ, Whitt DB, Brett G, Bryan FO, Feloy K, Long MC (2021). "The Impact of Climate Change on Ocean Submesoscale Activity." *Journal of Geophysical Research: Oceans*, **126**(5). doi:10.1002/essoar.10503524.1.

Richards KJ, Whitt DB, Brett GJ (2020). "Climate change impact on submesoscale ROMS data." doi:10.5281/zenodo.4615129.

Examples

```
library(cpop)
library(pacman)
p_load(tidyr,ggplot2,dplyr)

data(wavenumber_spectra)

# take logs of variables
data <-  wavenumber_spectra %>%  mutate_all(log) %>% rename_all( ~ paste0("log_", .x))
head(data)

# reproduce figure 4 in "The Impact of Climate Change on Ocean Submesoscale
# Activity" - Richards and Whitt (2021)
data %>%
gather(variable,log_power_spectra,-log_wavenumber) %>%
ggplot(aes(x=log_wavenumber, y=log_power_spectra, colour=variable)) +
geom_line() + theme_bw()
```

Index