Package: crops (via r-universe)

October 13, 2026

Type Package

Title Changepoints for a Range of Penalties (CROPS)

Version 1.0.5

Date 2026-10-12

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

Description Implements the Changepoints for a Range of Penalties (CROPS) algorithm of Haynes et al. (2017) <doi:10.1080/10618600.2015.1116445> for finding all of the optimal segmentations for multiple penalty values over a continuous range.

License GPL

Imports sets, memoise, magrittr, methods, Rdpack

RdMacros Rdpack

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Daniel Grose [aut, cre], Paul Fearnhead [aut], Idris Eckley [ctb]

Repository https://cran.r-universe.dev

Date/Publication 2026-10-12 11:20:02 UTC

RemoteUrl https://github.com/cran/crops

RemoteRef HEAD

RemoteSha 2a383e6d7f13d2b418c197c911b8b638e87ba009

Contents

crops 2
crops.class 3
crops.class-class 4

Index **5**

`crops` *Generic implementation of the crops algorithm (ref goes here).*

Description

Provides a generic implementation of the crops (changepoints for a range of penalties) algorithm of Haynes et al. (2014) which efficiently searches a range of penalty values in multiple changepoint problems. The crops algorithm finds the optimal segmentations for a different number of segments without incurring as large a computational cost as solving the constrained optimisation problem for a range of values for the number of changepoints. To make the method generic, the user must provide a function that maps a penalty value to the results obtained by a penalised cost changepoint method, and formats these results in a specific way. This interface to the generic method is similar to that as used by the **optimx** package.

Usage

```
crops(method, beta_min, beta_max, ...)
```

Arguments

`method`	A function mapping a penalty value to the results obtained by a penalised cost changepoint method. The function must return a list containing the cost and number of changepoints corresponding to the optimal segmentation as determined by a penalised cost changepoint method. Additional items can be appended to list which get propagated through the call to `crops`
`beta_min`	A positive numeric value indicating the smallest penalty value to consider.
`beta_max`	A positive numeric value indicating the maximum penalty value to consider.
`...`	Additional parameters to pass to the underlying changepoint method if required.

Value

An instance of an S4 class of type `crops.class`.

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.

Nash JC, Varadhan R (2011). "Unifying Optimization Algorithms to Aid Software System Users: optimx for R." *Journal of Statistical Software*, **43**(9), 1–14. https://www.jstatsoft.org/v43/i09/.

Nash JC (2014). "On Best Practice Optimization Methods in R." *Journal of Statistical Software*, **60**(2), 1–14. https://www.jstatsoft.org/v60/i02/.

Nash JC (2021). *optimx: Expanded Replacement and Extension of the 'optim' Function*. R package version 2021-6.12.

Maidstone R, Hocking T, Rigaill G, Fearnhead P (2017). "On optimal multiple changepoint algorithms for large data." *Statistics and Computing*, **27**. https://link.springer.com/article/10.1007/s11222-016-9636-3.

Rigaill G (2019). *fpop: Segmentation using Optimal Partitioning and Function Pruning*. R package version 2019.08.26.

Examples

```
library(crops)

# create a segmentation function for demonstration
toy_segmentation_function <- function(beta)
{
   x <- c(0.0,1.0,1.4,1.5,1.9)
   C <- c(2.5,4.0,4.4,5.6,9.5)
   n <- c(10,8,8,7,6)
   i <- findInterval(beta,x)
   return(as.list(c(C[i],n[i])))
}

# plot the segmentation function
beta <- seq(0.0,3.0,0.01)
values <- Map(toy_segmentation_function,beta)
C <- as.numeric(Map(function(.) (.)[[1]],values))
n <- as.numeric(Map(function(.) (.)[[2]],values))
plot(beta,n,type="l",col="red",ylim=c(2.0,10))
lines(beta,C,type="l",col="green")
legend("bottomright",
       legend = c("Unpenalized Cost", "Number of Changepoints"),
       col = c("green", "red"),
       pch = c(19, 17),
       lty = c(1, 1))

# analyse with crops
results <- crops(toy_segmentation_function,0.0,3.0)

# x summarise results
Map(results@method,as.list(results@betas))
```

`crops.class` *crops.class*

Description

Constructs a `crops.class` S4 class instance

Usage

```
crops.class(method, betas)
```

Arguments

`method`	A memoised function meeting the requirements of the crops function.
`betas`	A set of penalty values

Details

Constructor for the crops.class S4 class

Value

An instance of an S4 class of type `crops.class`

`crops.class-class` *crops.class*

Description

An S4 class to encapsulate results produced by running a `crops` analysis

Slots

`method` A memoised version of the segmentation function provided by the user

`betas` A set containing the beta values for which the sementation method was called during the `crops` analysis. These can be used in conjuction with `method` to efficiently recall the individual changepoint analysis results produced during the `crops` analysis

Index

.crops.class *(*crops.class-class*)*, 4

crops, 2, *4*
crops.class, 3, *4*
crops.class-class, 4