Package ‘capushe’

September 10, 2025

Type Package

Title Capushe, Data-Driven Slope Estimation and Dimension Jump

Version 1.1.3

Date 2025-09-06

Maintainer Vincent Brault <vincent.brault@univ-grenoble-alpes.fr>

Description Calibration of penalized criteria for model selection. The calibration methods available in this package are based on the slope heuristics.

Imports graphics, MASS, stats, methods, utils, grDevices

Collate capushe-package.R prog.R DDSE.R Djump.R capushe.R data.R

License GPL (>= 2.0)

Encoding UTF-8

LazyLoad yes

RoxygenNote 7.3.3

NeedsCompilation no

Author Vincent Brault [cre, aut] (ORCID: <https://orcid.org/0000-0002-3629-3429>),
Sylvain Arlot [ctb],
Jean-Patrick Baudry [ctb],
Cathy Maugis [ctb],
Bertrand Michel [ctb]

Repository CRAN

Date/Publication 2025-09-10 07:51:13 UTC

Contents

AICcapushe 2
BICcapushe 3
capushe 4
Capushe-class 5
datacapushe 6
datapartialcapushe 7

datavalidcapushe . . . 8
DDSE . . . 9
Djump . . . 11
plot-methods . . . 14
validation . . . 15

Index **17**

`AICcapushe` *AICcapushe*

Description

These functions return the model selected by the Akaike Information Criterion (AIC).

Usage

```
AICcapushe(data,n)
```

Arguments

`data` `data` is a matrix or a data.frame with four columns of the same length and each line corresponds to a model:

1. The first column contains the model names.
2. The second column contains the penalty shape values.
3. The third column contains the model complexity values.
4. The fourth column contains the minimum contrast value for each model.

`n` `n` is the sample size.

Details

The penalty shape value should be increasing with respect to the complexity value (column 3). The complexity values have to be positive. `n` is necessary to compute AIC and BIC criteria. `n` is the size of sample used to compute the contrast values given in the `data` matrix. Do not confuse `n` with the size of the model collection which is the number of rows of the `data` matrix.

Value

model The model selected by AIC.

Examples

```
data(datacapushe)
AICcapushe(datacapushe,n=1000)
```

BICcapushe *BICcapushe*

Description

These functions return the model selected by the Bayesian Information Criterion (BIC).

Usage

```
BICcapushe(data,n)
```

Arguments

`data` — `data` is a matrix or a data.frame with four columns of the same length and each line corresponds to a model:

1. The first column contains the model names.
2. The second column contains the penalty shape values.
3. The third column contains the model complexity values.
4. The fourth column contains the minimum contrast value for each model.

`n` — `n` is the sample size.

Details

The penalty shape value should be increasing with respect to the complexity value (column 3). The complexity values have to be positive. `n` is necessary to compute AIC and BIC criteria. `n` is the size of sample used to compute the contrast values given in the `data` matrix. Do not confuse `n` with the size of the model collection which is the number of rows of the `data` matrix.

Value

model The model selected by BIC.

Examples

```
data(datacapushe)
BICcapushe(datacapushe,n=1000)
```

capushe — *CAlibrating Penalities Using Slope HEuristics (CAPUSHE)*

Description

The `capushe` function proposes two algorithms based on the slope heuristics to calibrate penalties in the context of model selection via penalization.

Usage

```
capushe(data,n=0,pct=0.15,point=0,psi.rlm=psi.bisquare,scoef=2,Careajump=0,Ctresh=0)
```

Arguments

`data`	`data` is a matrix or a data.frame with four columns of the same length and each line corresponds to a model: 1. The first column contains the model names. 2. The second column contains the penalty shape values. 3. The third column contains the model complexity values. 4. The fourth column contains the minimum contrast value for each model.
`n`	`n` is the sample size.
`pct`	Minimum percentage of points for the plateau selection. See `DDSE` for more details.
`point`	Minimum number of point for the plateau selection (See `DDSE` for more details). If `point` is different from 0, `pct` is obsolete.
`psi.rlm`	Weight function used by `rlm`. See `DDSE` for more details. `psi.rlm`="lm" for non robust linear regression.
`scoef`	Ratio parameter. Default value is 2.
`Careajump`	Constant of jump area (See `Djump` for more details). Default value is 0 (no area).
`Ctresh`	Maximal treshold for the complexity associated to the penalty coefficient (See `Djump` for more details). Default value is 0 (Maximal jump selected as the greater jump).

Details

The model $\hat{m}$ selected by the procedure fulfills $\hat{m} = \text{argmin}\ \gamma_n(\hat{s}_m) + scoef \times \kappa \times pen_{shape}(m)$ where

- κ is the penalty coefficient.
- γ_n is the empirical contrast.
- $\hat{s}_m$ is the estimator for the model m.
- $scoef$ is the ratio parameter.
- pen_{shape} is the penalty shape.

The capushe function calls the functions `DDSE` and `Djump` to calibrate κ, see the description of these functions for more details. In the case of equality between two penalty shape values, only the model with the smallest contrast is considered.

Author(s)

Vincent Brault

References

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

See Also

`Djump`, `DDSE`, `AIC` or `BIC` to use only one of these model selection functions. `plot` for graphical displays of DDSE and Djump.

Examples

```
data(datacapushe)
capushe(datacapushe)
capushe(datacapushe,1000)
```

`Capushe-class` *Class "Capushe"*

Description

Class of object returned by the `capushe` function.

Arguments

`object`	an object with class `capushe`

Slots

`DDSE` A list returned by the `DDSE` function.

`Djump` A list returned by the `Djump` function.

`AIC_capushe` A list returned by the `AICcapushe` function.

`BIC_capushe` A list returned by the `BICcapushe` function.

`n` The number of observations given by the user.

References

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

See Also

See also `plot,Capushe-method` and `capushe`.

datacapushe *datacapushe*

Description

A dataframe example for the capushe package based on a simulated Gaussian mixture dataset in R^3.

Usage

```
data(datacapushe)
```

Format

A data frame with 50 rows (models) and the following 4 variables:

`model` a character vector: model names.

`pen` a numeric vector: model penalty shape values.

`complexity` a numeric vector: model complexity values.

`contrast` a numeric vector: model contrast values.

Details

The simulated dataset is composed of $n = 1000$ observations in R^3. It consists of an equiprobable mixture of three large "bubble" groups centered at $\nu_1 = (0,0,0)$, $\nu_2 = (6,0,0)$ and $\nu_3 = (0,6,0)$ respectively. Each bubble group j is simulated from a mixture of seven components according to the following density distribution:

$x \in \mathsf{R}^3 \to 0.4\Phi(x|\mu_1 + \nu_j, I_3) + \sum_{k=2}^{7} 0.1\Phi(x|\mu_k + \nu_j, 0.1I_3)$

with $\mu_1 = (0,0,0)$, $\mu_2 = (0,0,1.5)$, $\mu_3 = (0,1.5,0)$, $\mu_4 = (1.5,0,0,)$, $\mu_5 = (0,0,-1.5)$, $\mu_6 = (0,-1.5,0)$ and $\mu_7 = (-1.5,0,0,)$. Thus the distribution of the dataset is actually a 21-component Gaussian mixture.

A model collection of spherical Gaussian mixtures is considered and the dataframe `datacapushe` contains the maximum likelihood estimations for each of these models. The number of free parameters of each model is used for the complexity values and pen_{shape} is defined by this complexity divided by n.

`datapartialcapushe` and `datavalidcapushe` can be used to run the `validation` function. `datapartialcapushe` only contains the models with less than 21 components. `datavalidcapushe` contains three models with 30, 40 and 50 components respectively.

References

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

Examples

```
data(datacapushe)
capushe(datacapushe,n=1000)
## BIC, DDSE and Djump all three select the true model
plot(capushe(datacapushe),newwindow=FALSE)
## Validation:
data(datapartialcapushe)
capushepartial=capushe(datapartialcapushe)
data(datavalidcapushe)
validation(capushepartial,datavalidcapushe,newwindow=FALSE) ## The slope heuristics should not
## be applied for datapartialcapushe.
```

`datapartialcapushe` *datapartialcapushe*

Description

A dataframe example for the capushe package based on a simulated Gaussian mixture dataset in R^3.

Usage

```
data(datapartialcapushe)
```

Format

A data frame with 21 rows (models) and the following 4 variables:

`model` a character vector: model names.

`pen` a numeric vector: model penalty shape values.

`complexity` a numeric vector: model complexity values.

`contrast` a numeric vector: model contrast values.

Details

The simulated dataset is composed of $n = 1000$ observations in R^3. It consists of an equiprobable mixture of three large "bubble" groups centered at $\nu_1 = (0,0,0)$, $\nu_2 = (6,0,0)$ and $\nu_3 = (0,6,0)$ respectively. Each bubble group j is simulated from a mixture of seven components according to the following density distribution:

$x \in \mathsf{R}^3 \rightarrow 0.4\Phi(x|\mu_1 + \nu_j, I_3) + \sum_{k=2}^{7} 0.1\Phi(x|\mu_k + \nu_j, 0.1I_3)$

with $\mu_1 = (0,0,0)$, $\mu_2 = (0,0,1.5)$, $\mu_3 = (0,1.5,0)$, $\mu_4 = (1.5,0,0,)$, $\mu_5 = (0,0,-1.5)$, $\mu_6 = (0,-1.5,0)$ and $\mu_7 = (-1.5,0,0,)$. Thus the distribution of the dataset is actually a 21-component Gaussian mixture.

A model collection of spherical Gaussian mixtures is considered and the dataframe datacapushe contains the maximum likelihood estimations for each of these models. The number of free parameters of each model is used for the complexity values and pen_{shape} is defined by this complexity divided by n.

datapartialcapushe and datavalidcapushe can be used to run the validation function. datapartialcapushe only contains the models with less than 21 components. datavalidcapushe contains three models with 30, 40 and 50 components respectively.

References

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

Examples

```
data(datacapushe)
capushe(datacapushe,n=1000)
## BIC, DDSE and Djump all three select the true model
plot(capushe(datacapushe),newwindow=FALSE)
## Validation:
data(datapartialcapushe)
capushepartial=capushe(datapartialcapushe)
data(datavalidcapushe)
validation(capushepartial,datavalidcapushe,newwindow=FALSE) ## The slope heuristics should not
## be applied for datapartialcapushe.
```

---

datavalidcapushe *datavalidcapushe*

Description

A dataframe example for the capushe package based on a simulated Gaussian mixture dataset in R^3.

Usage

```
data(datavalidcapushe)
```

Format

A data frame with 3 rows (models) and the following 4 variables:

- model a character vector: model names.
- pen a numeric vector: model penalty shape values.
- complexity a numeric vector: model complexity values.
- contrast a numeric vector: model contrast values.

Details

The simulated dataset is composed of $n = 1000$ observations in R^3. It consists of an equiprobable mixture of three large "bubble" groups centered at $\nu_1 = (0,0,0)$, $\nu_2 = (6,0,0)$ and $\nu_3 = (0,6,0)$ respectively. Each bubble group j is simulated from a mixture of seven components according to the following density distribution:

$x \in \mathsf{R}^3 \to 0.4\Phi(x|\mu_1 + \nu_j, I_3) + \sum_{k=2}^{7} 0.1\Phi(x|\mu_k + \nu_j, 0.1I_3)$

with $\mu_1 = (0,0,0)$, $\mu_2 = (0,0,1.5)$, $\mu_3 = (0,1.5,0)$, $\mu_4 = (1.5,0,0,)$, $\mu_5 = (0,0,-1.5)$, $\mu_6 = (0,-1.5,0)$ and $\mu_7 = (-1.5,0,0,)$. Thus the distribution of the dataset is actually a 21-component Gaussian mixture.

A model collection of spherical Gaussian mixtures is considered and the dataframe `datacapushe` contains the maximum likelihood estimations for each of these models. The number of free parameters of each model is used for the complexity values and pen_{shape} is defined by this complexity divided by n.

`datapartialcapushe` and `datavalidcapushe` can be used to run the `validation` function. `datapartialcapushe` only contains the models with less than 21 components. `datavalidcapushe` contains three models with 30, 40 and 50 components respectively.

References

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

Examples

```
data(datacapushe)
capushe(datacapushe,n=1000)
## BIC, DDSE and Djump all three select the true model
plot(capushe(datacapushe),newwindow=FALSE)
## Validation:
data(datapartialcapushe)
capushepartial=capushe(datapartialcapushe)
data(datavalidcapushe)
validation(capushepartial,datavalidcapushe,newwindow=FALSE) ## The slope heuristics should not
## be applied for datapartialcapushe.
```

`DDSE` *Model selection by Data-Driven Slope Estimation*

Description

`DDSE` is a model selection function based on the slope heuristics.

Usage

```
DDSE(data, pct = 0.15, point = 0, psi.rlm = psi.bisquare, scoef = 2)
```

Arguments

data	data is a matrix or a data.frame with four columns of the same length and each line corresponds to a model: 1. The first column contains the model names. 2. The second column contains the penalty shape values. 3. The third column contains the model complexity values. 4. The fourth column contains the minimum contrast value for each model.
pct	Minimum percentage of points for the plateau selection. It must be between 0 and 1. Default value is 0.15.
point	Minimum number of point for the plateau selection. If point is different from 0, pct is obsolete.
psi.rlm	Weight function used by rlm. psi.rlm="lm" for non robust linear regression.
scoef	Ratio parameter. Default value is 2.

Details

Let M be the model collection and $P = \{pen_{shape}(m), m \in M\}$. The DDSE algorithm proceeds in four steps:

1. If several models in the collection have the same penalty shape value (column 2), only the model having the smallest contrast value $\gamma_n(\hat{s}_m)$ (column 4) is considered.
2. For any $p \in P$, the slope $\hat{\kappa}(p)$ (argument @kappa) of the linear regression (argument psi.rlm) on the couples of points $\{(pen_{shape}(m), -\gamma_n(\hat{s}_m)); pen_{shape}(m) \geq p\}$ is computed.
3. For any $p \in P$, the model fulfilling the following condition is selected: $\hat{m}(p) =$ argmin $\gamma_n(\hat{s}_m) + scoef \times \hat{\kappa}(p) \times pen_{shape}(m)$. This gives an increasing sequence of change-points $(p_i)_{1 \leq i \leq I+1}$ (output @ModelHat$point_breaking). Let $(N_i)_{1 \leq i \leq I}$ (output @ModelHat$number_plateau) be the lengths of each "plateau".
4. If point is different from 0, let $\hat{i} =$ max $\{1 \leq i \leq I; N_i \geq point\}$ else let $\hat{i} =$ max $\{1 \leq i \leq I; N_i \geq pct \sum_{l=1}^{I} N_l\}$ (output @ModelHat$imax). The model $\hat{m}(p_{\hat{i}})$ (output @model) is finally returned.

The "slope interval" is the interval $[a, b]$ where $a = inf\{\hat{\kappa}(p), p \in [p_{\hat{i}}, p_{\hat{i}+1}[\cap P\}$ and $b = sup\{\hat{\kappa}(p), p \in [p_{\hat{i}}, p_{\hat{i}+1}[\cap P\}$.

Value

@model	The model selected by the DDSE algorithm.
@kappa	The vector of the successive slope values.
@ModelHat	A list describing the algorithm.
@ModelHat$model_hat	The vector of preselected models $\hat{m}(p)$.
@ModelHat$point_breaking	The vector of the breaking points $(p_i)_{1 \leq i \leq I+1}$.
@ModelHat$number_plateau	The vector of the lengths $(N_i)_{1 \leq i \leq I}$.

`@ModelHat$imax`	The rank $\hat{i}$ of the selected plateau.
`@interval`	A list about the "slope interval".
`@interval$interval`	The slope interval.
`@interval$percent_of_points`	The proportion $N_{\hat{i}}/\sum_{l=1}^{I} N_l$.
`@graph`	A list computed for the `plot` method.

Author(s)

Vincent Brault

References

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

See Also

`capushe` for a model selection function including `AIC`, `BIC`, the `DDSE` algorithm and the `Djump` algorithm. `plot` for graphical dsiplays of the `DDSE` algorithm and the `Djump` algorithm.

`Djump` *Model selection by dimension jump*

Description

`Djump` is a model selection function based on the slope heuristics.

Usage

```
Djump(data,scoef=2,Careajump=0,Ctresh=0)

Djump(data, scoef = 2, Careajump = 0, Ctresh = 0)
```

Arguments

`data` — `data` is a matrix or a data.frame with four columns of the same length and each line corresponds to a model:

1. The first column contains the model names.
2. The second column contains the penalty shape values.
3. The third column contains the model complexity values.
4. The fourth column contains the minimum contrast value for each model.

`scoef` — Ratio parameter. Default value is 2.

`Careajump` — Constant of jump area (See `Djump` for more details). Default value is 0 (no area).

`Ctresh` — Maximal treshold for the complexity associated to the penalty coefficient (See `Djump` for more details). Default value is 0 (Maximal jump selected as the greater jump).

Details

`Djump` is a model selection function based on the slope heuristics.

The Djump algorithm proceeds in three steps:

1. For all $\kappa > 0$, compute $m(\kappa) \in argmin_{m \in M}\{\gamma_n(\hat{s}_m) + \kappa \times pen_{shape}(m)\}$ This gives a decreasing step function $\kappa \mapsto C_{m(\kappa)}$.
2. Find $\hat{\kappa}$ such that $C_{m(\hat{\kappa})}$ corresponds to the greatest jump of complexity if $C_{tresh} = 0$ else $\hat{\kappa}$ such that $\hat{\kappa} = inf\{\kappa > 0 : C_{m(\kappa)} \leq C_{tresh}\}$.
3. Select $\hat{m} = m(scoef \times \hat{\kappa})$ (output `@model`).

Arlot has proposed a jump area containing the maximal jump defined by : $[\kappa(1-Careajump); \kappa(1+Careajump)]$. If $Careajump > 0$, `Djump` return the area with the greatest jump. In practice, it is advisable to take $Careajump = \frac{log(n)}{n}$ where n is the number of observations.

The Djump algorithm proceeds in three steps:

1. For all $\kappa > 0$, compute $m(\kappa) \in argmin_{m \in M}\{\gamma_n(\hat{s}_m) + \kappa \times pen_{shape}(m)\}$ This gives a decreasing step function $\kappa \mapsto C_{m(\kappa)}$.
2. Find $\hat{\kappa}$ such that $C_{m(\hat{\kappa})}$ corresponds to the greatest jump of complexity if $C_{tresh} = 0$ else $\hat{\kappa}$ such that $\hat{\kappa} = inf\{\kappa > 0 : C_{m(\kappa)} \leq C_{tresh}\}$.
3. Select $\hat{m} = m(scoef \times \hat{\kappa})$ (output `@model`).

Arlot has proposed a jump area containing the maximal jump defined by : $[\kappa(1-Careajump); \kappa(1+Careajump)]$. If $Careajump > 0$, `Djump` return the area with the greatest jump. In practice, it is advisable to take $Careajump = \frac{log(n)}{n}$ where n is the number of observations.

Value

`@model` The `model` selected by the dimension jump method.

`@ModelHat` A list describing the algorithm.

`@ModelHat$jump` The vector of jump heights.

`@ModelHat$kappa` The vector of the values of κ at each jump.

`@ModelHat$model_hat` The vector of the selected models $m(\kappa)$ by the jump.

`@ModelHat$JumpMax` The location of the greatest jump.

`@ModelHat$Kopt` $\kappa_{opt} = scoef\hat{\kappa}$.

`@graph` A list computed for the `plot` method.

`@model` The `model` selected by the dimension jump method.

`@ModelHat` A list describing the algorithm.

`@ModelHat$jump` The vector of jump heights.

`@ModelHat$kappa` The vector of the values of κ at each jump.

`@ModelHat$model_hat` The vector of the selected models $m(\kappa)$ by the jump.

`@ModelHat$JumpMax` The location of the greatest jump.

`@ModelHat$Kopt` $\kappa_{opt} = scoef\hat{\kappa}$.

`@graph` A list computed for the `plot` method.

Slots

`model` character. The `model` selected by the dimension jump method.

`ModelHat` list. A list describing the algorithm.

- `jump` The vector of jump heights.
- `kappa` The vector of the values of κ at each jump.
- `model_hat` The vector of the selected models $m(\kappa)$ by the jump.
- `JumpMax` The location of the greatest jump.
- `Kopt` $\kappa_{opt} = scoef\hat{\kappa}$.

`graph` list.

`Area` list.

`graph` list.

`Area` list.

Author(s)

Vincent Brault

References

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

See Also

`capushe` for a model selection function including `AIC`, `BIC`, the `DDSE` algorithm and the `Djump` algorithm. `plot` for a graphical display of the `DDSE` algorithm and the `Djump` algorithm.

`capushe` for a model selection function including `AIC`, `BIC`, the `DDSE` algorithm and the `Djump` algorithm. `plot` for a graphical display of the `DDSE` algorithm and the `Djump` algorithm.

Examples

```
data(datacapushe)
Djump(datacapushe)
res <- Djump(datacapushe)
plot(res,newwindow=FALSE)
res <- Djump(datacapushe,Careajump=sqrt(log(1000)/1000))
plot(res,newwindow=FALSE)
res <- Djump(datacapushe,Ctresh=1000/log(1000))
plot(res,newwindow=FALSE)
data(datacapushe)
Djump(datacapushe)
plot(Djump(datacapushe),newwindow=FALSE)
Djump(datacapushe,Careajump=sqrt(log(1000)/1000))
plot(Djump(datacapushe,Careajump=sqrt(log(1000)/1000)),newwindow=FALSE)
Djump(datacapushe,Ctresh=1000/log(1000))
plot(Djump(datacapushe,Ctresh=1000/log(1000)),newwindow=FALSE)
```

plot-methods *Plot for capushe*

Description

The plot methods allow the user to check that the slope heuristics can be applied confidently.

- `signature(x = "Capushe")` This graphical function displays the `DDSE` plot and the `Djump` plot.
- `signature(x = "DDSE")` This graphical function displays the `DDSE` plot.
- `signature(x = "Djump")` This graphical function displays the `Djump` plot.

Usage

```
plot(x,y, ...)
```

Arguments

`x`	Output of `DDSE`, `Djump` or `capushe`.
`...`	other arguments : • newwindow If `newwindow=TRUE` (default value), a new window is created for each plot. • ask If `ask=TRUE` (default value), `plot` waits for the user to press a key to display the next plot (only for the class `capushe`).
`y`	is unused.

Details

The graphical window of `DDSE` is composed of three graphics (see `DDSE` for more details):

left The left plot shows $-\gamma_n(\hat{s}_m)$ with respect to the penalty shape values.

topright Successive slope values $\hat{\kappa}(p)$.

bottomright The bottomright plot shows the selected models $\hat{m}(p)$ with respect to the successive slope values. The plateau in blue is selected.

The graphical window of `Djump` shows the complexity $C_{m(\kappa)}$ of the selected model with respect to κ. $\hat{\kappa}^{dj}$ corresponds to the greatest jump. κ_{opt} is defined by $\kappa_{opt} = scoef \times \hat{\kappa}^{dj}$. The red line represents the slope interval computed by the `DDSE` algorithm (only for `capushe`). See `Djump` for more details.

Note

Use `newwindow=FALSE` to produce a PDF files (for an object of class `capushe`, use moreover `ask=FALSE`).

`validation` *validation*

Description

`validation` checks that the slope heuristics can be applied confidently.

Usage

```
validation(x,data2,...)
```

Arguments

`x`	`x` must be an object of `class` `capushe` or `DDSE`, in practice an output of the `capushe` function or the `DDSE` function.
`data2`	`data2` is a matrix or a data.frame with four columns of the same length and each line corresponds to a model: 1. The first column contains the model names. 2. The second column contains the penalty shape values. 3. The third column contains the model complexity values. 4. The fourth column contains the minimum contrast value for each model.
`...`	• If `newwindow==TRUE`, a new window is created for the plot.

Details

The `validation` function plots the additional and more complex models `data2` to check that the linear relation between the penalty shape values and the contrast values (which is recorded in `x`) is valid for the more complex models.

Author(s)

Brault Vincent

References

Article: Baudry, J.-P., Maugis, C. and Michel, B. (2011) Slope heuristics: overview and implementation. *Statistics and Computing*, to appear. doi: 10.1007/s11222-011-9236-1

See Also

`capushe` for a more general model selection function including `AIC`, `BIC`, the `DDSE` algorithm and the `Djump` algorithm.

Examples

```
data(datapartialcapushe)
capushepartial=capushe(datapartialcapushe)
data(datavalidcapushe)
validation(capushepartial,datavalidcapushe,newwindow=FALSE) ## The slope heuristics should not
## be applied for datapartialcapushe.
data(datacapushe)
plot(capushe(datacapushe),newwindow=FALSE)
```

Index

∗ **datasets**
datacapushe, 6
datapartialcapushe, 7
datavalidcapushe, 8
∗ **methods**
plot-methods, 14
∗ **models**
DDSE, 9
validation, 15

AIC, *5*, *11*, *13*, *15*
AICcapushe, 2, *5*

BIC, *5*, *11*, *13*, *15*
BICcapushe, 3, *5*

CAlibrating *(capushe)*, 4
Calibrating *(capushe)*, 4
CAPUSHE *(capushe)*, 4
Capushe *(capushe)*, 4
capushe, 4, *5*, *11*, *13–15*
Capushe-class, 5

Data-Driven *(DDSE)*, 9
Data-driven *(DDSE)*, 9
data-driven *(DDSE)*, 9
datacapushe, 6
datapartialcapushe, 7
datavalidcapushe, 8
DDSE, *4, 5*, 9, *13–15*
Ddse *(DDSE)*, 9
ddse *(DDSE)*, 9
DDSE-class *(DDSE)*, 9
Dimension_Jump *(Djump)*, 11
Dimension_jump *(Djump)*, 11
DimensionJump *(Djump)*, 11
Dimensionjump *(Djump)*, 11
DJUMP *(Djump)*, 11
Djump, *4, 5*, *11*, 11, *14, 15*
djump *(Djump)*, 11
Djump-class *(Djump)*, 11

plot, *5*, *11–13*
plot *(plot-methods)*, 14
plot,Capushe-method *(plot-methods)*, 14
plot,DDSE-method *(plot-methods)*, 14
plot,Djump-method *(plot-methods)*, 14
plot-methods, 14
plot.capushe *(plot-methods)*, 14
plot.DDSE *(plot-methods)*, 14
plot.Djump *(plot-methods)*, 14
plotcapushe *(plot-methods)*, 14
plotDDSE *(plot-methods)*, 14
plotDjump *(plot-methods)*, 14
print,Capushe-method *(capushe)*, 4
print,DDSE-method *(DDSE)*, 9
print,Djump-method *(Djump)*, 11
print.capushe *(capushe)*, 4
print.DDSE *(DDSE)*, 9
print.Djump *(Djump)*, 11

rlm, *4*, *10*

show,Capushe-method *(capushe)*, 4
show,DDSE-method *(DDSE)*, 9
show,Djump-method *(Djump)*, 11
show.capushe *(capushe)*, 4
show.DDSE *(DDSE)*, 9
show.Djump *(Djump)*, 11
summary,Capushe-method *(capushe)*, 4
summary,DDSE-method *(DDSE)*, 9
summary,Djump-method *(Djump)*, 11
summary.capushe *(capushe)*, 4
summary.DDSE *(DDSE)*, 9
summary.Djump *(Djump)*, 11

validation, *6*, *8, 9*, 15
validation,Capushe-method *(capushe)*, 4
validation,DDSE-method *(DDSE)*, 9
validation-methods *(validation)*, 15

validation.capushe *(validation)*, 15
validation.DDSE *(validation)*, 15
validationcapushe *(validation)*, 15
validationDDSE *(validation)*, 15