

Package ‘oscar’

July 22, 2025

Type Package

Title Optimal Subset Cardinality Regression (OSCAR) Models Using the L0-Pseudonorm

Version 1.2.1

Date 2023-10-02

Description Optimal Subset Cardinality Regression (OSCAR) models offer regularized linear regression using the L0-pseudonorm, conventionally known as the number of non-zero coefficients. The package estimates an optimal subset of features using the L0-penalization via cross-validation, bootstrapping and visual diagnostics. Effective Fortran implementations are offered along the package for finding optima for the DC-decomposition, which is used for transforming the discrete L0-regularized optimization problem into a continuous non-convex optimization task. These optimization modules include DBDC ('Double Bundle method for nonsmooth DC optimization' as described in Joki et al. (2018) <[doi:10.1137/16M1115733](https://doi.org/10.1137/16M1115733)>) and LMBM ('Limited Memory Bundle Method for large-scale nonsmooth optimization' as in Haarala et al. (2004) <[doi:10.1080/10556780410001689225](https://doi.org/10.1080/10556780410001689225)>). The OSCAR models are comprehensively exemplified in Halkola et al. (2023) <[doi:10.1371/journal.pcbi.1010333](https://doi.org/10.1371/journal.pcbi.1010333)>). Multiple regression model families are supported: Cox, logistic, and Gaussian.

License GPL-3

LazyData true

URL <https://github.com/Syksy/oscar>

BugReports <https://github.com/Syksy/oscar/issues>

NeedsCompilation yes

Depends R (>= 3.6.0)

Imports graphics, grDevices, hamlet, Matrix, methods, stats, survival, utils, pROC

Suggests ePCR, glmnet, knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

Author Teemu Daniel Laajala [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7016-7354>),
 Kaisa Joki [aut],
 Anni Halkola [aut]

Maintainer Teemu Daniel Laajala <teelaa@utu.fi>

Repository CRAN

Date/Publication 2023-10-02 13:40:08 UTC

Contents

oscar-package	3
coef,oscar-method	3
cost	4
Example data	5
feat	5
kits	6
oscar	6
oscar-class	8
oscar.binarize	8
oscar.binplot	9
oscar.bs	10
oscar.bs.boxplot	11
oscar.bs.k	12
oscar.bs.plot	12
oscar.bs.visu	14
oscar.control	14
oscar.cost.after	17
oscar.cv	17
oscar.cv.visu	18
oscar.pareto	19
oscar.pareto.visu	20
oscar.sparsify	21
oscar.visu	22
show,oscar-method	23

Index	24
--------------	-----------

oscar-package

oscar: Optimal Subset Cardinality Regression

Description

OSCAR models utilize the L0-pseudonorm to select an optimal subset of features that generalizes linear regression models to a variety of families. Currently supported models include conventional Gaussian regression (family="mse" or family="gaussian"), Binomial/Logistic regression (family="logistic"), and Cox proportional hazards modeling (family="cox").

References

Halkola AS, Joki K, Mirtti T, Mäkelä MM, Aittokallio T, Laajala TD (2023) OSCAR: Optimal subset cardinality regression using the L0-pseudonorm with applications to prognostic modelling of prostate cancer. PLoS Comput Biol 19(3): e1010333. doi:10.1371/journal.pcbi.1010333

coef, oscar-method

Extract coefficients of oscar-objects

Description

Extract coefficients of oscar-objects

Prediction based on oscar-objects

Plot oscar-coefficients as a function of k and override default plot generic

Usage

```
## S4 method for signature 'oscar'
coef(object, k)

## S4 method for signature 'oscar'
predict(
  object,
  k,
  type = c("response", "link", "nonzero", "coefficients", "label"),
  newdata = object@x
)

## S4 method for signature 'oscar'
plot(x, y, k = 1:x@kmax, add = FALSE, intercept = FALSE, ...)
```

Arguments

object	Fit oscar S4-object
k	Vector of cardinality 'k' values
type	Type of prediction; valid values are 'response', 'link', 'nonzero', 'coefficients', or 'label'
newdata	Data to predict on; if no alternate is supplied, the function uses the original 'x' data matrix used to fit object
x	Values on x-axis
y	Values on y-axis
add	Should the plot be added on top of an existing plot (if FALSE, create a new graphics device), Default: FALSE
intercept	Should model intercept be plotted, Default: FALSE
...	Additional parameters passed on to the points-function drawing lines as a function of cardinality

Value

Vector of model coefficient values at given cardinality 'k'

A vector of coefficient predictions at the specified cardinality 'k' with a format depending on the supplied 'type' parameter

Override default plot function with no return but instead tailor suitable graphics plotting

cost	<i>Return total cost of model fit based on provided kit/variable costs vector</i>
------	---

Description

Return total cost of model fit based on provided kit/variable costs vector

Return total cost of model fit based on provided kit/variable costs vector

Usage

```
cost(object, k)
```

```
## S4 method for signature 'oscar'
cost(object, k)
```

Arguments

object	Fit oscar S4-object
k	Cardinality 'k' to compute total feature cost at

Value

Numeric value of total feature/kit cost at cardinality 'k'
Numeric value of total feature/kit cost at cardinality 'k'

Example data	<i>Example data from TYKS / HUSLAB</i>
--------------	--

Description

An example data set from mCRPC patients in TYKS, along with cost vector / kit structure from HUSLAB

Usage

data(ex)

Examples

data(ex)

feat	<i>Return named vector of feature indices with a given k that are non-zero</i>
------	--

Description

Return named vector of feature indices with a given k that are non-zero
Return named vector of feature indices with a given k that are non-zero

Usage

feat(object, k)

S4 method for signature 'oscar'
feat(object, k)

Arguments

object Fit oscar S4-object
k Cardinality 'k' to extract non-zero features at

Value

Vector of feature indices at cardinality 'k'
Vector of feature indices at cardinality 'k'

kits	<i>Return named vector of indices for kits with a given k that are non-zero</i>
------	---

Description

Return named vector of indices for kits with a given k that are non-zero

Return named vector of indices for kits with a given k that are non-zero

Usage

```
kits(object, k)
```

```
## S4 method for signature 'oscar'
```

```
kits(object, k)
```

Arguments

object Fit oscar S4-object

k Cardinality 'k' to extract kit indices at

Value

Vector of kit indices at cardinality 'k'

Vector of kit indices at cardinality 'k'

oscar	<i>Main OSCAR fitting function</i>
-------	------------------------------------

Description

This function fits an OSCAR model object to the provided training data with the desired model family.

Usage

```
oscar(
  x,
  y,
  k,
  w,
  family = "cox",
  metric,
  solver = 1,
  verb = 1,
  print = 3,
```

```

    kmax,
    sanitize = TRUE,
    percentage = 1,
    in_selection = 1,
    storeX = TRUE,
    storeY = TRUE,
    control,
    ...
)

```

Arguments

x	Data matrix 'x'
y	Response vector/two-column matrix 'y' (see: family); number of rows equal to nrow(x)
k	Integer (0/1) kit indicator matrix; number of columns equal to ncol(x), Default: Unit diagonal indicator matrix
w	Kit cost weight vector w of length nrow(k), Default: Equal cost for all variables
family	Model family, should be one of: 'cox', 'mse'/'gaussian', or 'logistic', Default: 'cox'
metric	Goodness metric, Default(s): Concordance index for Cox, MSE for Gaussian, and AUC for logistic regression
solver	Solver used in the optimization, should be 1/'DBDC' or 2/'LMBM', Default: 1.
verb	Level of verbosity in R, Default: 1
print	Level of verbosity in Fortran (may not be visible on all terminals); should be an integer between range, range, Default: 3
kmax	Maximum k step tested, by default all k are tested from k to maximum dimensionality, Default: ncol(x)
sanitize	Whether input column names should be cleaned of potentially problematic symbols, Default: TRUE
percentage	Percentage of possible starting points used within range [0,1], Default: 1
in_selection	Which starting point selection strategy is used (1, 2 or 3), Default: 1
storeX	If data matrix X should be saved in the model object; turning this off might help with memory, Default: TRUE
storeY	If data response Y should be saved in the model object; turning this off might help with memory, Default: TRUE
control	Tuning parameters for the optimizers, see function oscar.control(), Default: see ?oscar.control
...	Additional parameters

Details

OSCAR utilizes the L0-pseudonorm, also known as the best subset selection, and makes use of a DC-formulation of the discrete feature selection task into a continuous one. Then an appropriate optimization algorithm is utilized to find optima at different cardinalities (k). The S4 model objects 'oscar' can then be passed on to various down-stream functions, such as `oscar.pareto`, `oscar.cv`, and `oscar.bs`, along with their supporting visualization functions.

Value

Fitted oscar-object

See Also

[oscar.cv](#) [oscar.bs](#) [oscar.pareto](#) [oscar.visu](#) [oscar.cv.visu](#) [oscar.bs.visu](#) [oscar.pareto.visu](#) [oscar.binplot](#)

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit
}
```

oscar-class	<i>S4-class for oscar</i>
-------------	---------------------------

Description

S4-class for oscar

<code>oscar.binarize</code>	<i>Binary logical indicator matrix representation of an oscar object's coefficients (zero vs. non-zero, i.e. feature inclusion)</i>
-----------------------------	---

Description

Create a sparse matrix with binary indicator 1 indicating that a coefficient was non-zero, and value 0 (or . in sparse matrix) indicating that a coefficient was zero (i.e. feature not included)

Usage

```
oscar.binarize(fit, kmax = fit@kmax)
```

Arguments

<code>fit</code>	Fit oscar-model object
<code>kmax</code>	Create matrix until kmax-value; by default same as for fit object, but for high dimensional tasks one may wish to reduce this

Details

The matrix consists of TRUE/FALSE values, and is very similar to the `oscar.sparsify`, where the function provides estimate values in a sparse matrix format.

Value

A binary logical indicator matrix of variables (rows) as a function of cardinality k (columns), where elements are binary indicators for 1 as non-zero and 0 as zero.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  oscar.binplot(fit, kmax=5)
}
```

<code>oscar.binplot</code>	<i>Visualize binary indicator matrix optionally coupled with cross-validation performance for oscar models</i>
----------------------------	--

Description

This visualization function makes use of the sparsified beta-coefficient matrix form as a function of cardinality. Optionally, user may showcase cross-validation performance alongside at the same cardinality values.

Usage

```
oscar.binplot(
  fit,
  cv,
  kmax,
  collines = TRUE,
  rowlines = TRUE,
  cex.axis = 0.6,
  heights = c(0.2, 0.8),
  ...
)
```

Arguments

<code>fit</code>	Fitted oscar S4-class object
<code>cv</code>	Matrix produced by <code>oscar.cv</code> ; rows are cv-folds, cols are k-values
<code>kmax</code>	Maximum cardinality 'k'
<code>collines</code>	Should vertical lines be drawn to bottom part
<code>rowlines</code>	Should horizontal lines be drawn to highlight variables
<code>cex.axis</code>	Axis magnification
<code>heights</code>	Paneling proportions as a numeric vector of length 2
<code>...</code>	Additional parameters passed on to <code>hamlet::hmap</code>

Value

This is a plotting function that does not return anything, but instead draws on a new graphics device.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_cv <- oscar.cv(fit, fold = 10, seed = 123)
  oscar.binplot(fit=fit, cv=fit_cv)
}
```

oscar.bs

Bootstrapping for oscar-fitted model objects

Description

This model bootstraps the fitting of a given oscar object (re-fits the model for data that is equal in size but sampled with replacement). The output objects give insight into robustness of the oscar-coefficient path, as well as relative importance of model objects.

Usage

```
oscar.bs(fit, bootstrap = 100, seed = NULL, verb = 0, ...)
```

Arguments

<code>fit</code>	oscar-model object
<code>bootstrap</code>	Number of bootstrapped datasets, Default: 100
<code>seed</code>	Random seed for reproducibility with NULL indicating that it is not set, Default: NULL
<code>verb</code>	Level of verbosity with higher integer giving more information, Default: 0
<code>...</code>	Additional parameters passed to oscar-function

Details

The function provides a fail-safe try-catch in an event of non-convergence of the model fitting procedure. This may occur for example if a bootstrapped data matrix has a column consist of a single value only over all observations.

Value

3-dimensional array with dimensions corresponding to k-steps, beta coefficients, and bootstrap runs

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_bs <- oscar.cv(fit, bootstrap = 20, seed = 123)
  fit_bs
}
```

oscar.bs.boxplot

Bootstrap visualization with boxplot, percentage of new additions

Description

This function plots as barplots as a function of k-cardinality in what properties certain coefficients were chosen as non-zero over the bootstrap runs.

Usage

```
oscar.bs.boxplot(bs, ...)
```

Arguments

bs	Bootstrapped 3-dimensional array for an oscar object as produced by oscar.bs
...	Additional parameters passed on to barplot

Value

This is a plotting function that does not return anything, but instead draws on a new graphics device.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_bs <- oscar.bs(fit, bootstrap = 20, seed = 123)
  oscar.bs.boxplot(fit_bs)
}
```

oscar.bs.k	<i>Reformatting bootstrap output for cardinality k rows</i>
------------	---

Description

The function reformats bootstrapped runs to a single long data.frame, where all bootstrapped runs are covered along with the choices for the variables at each cardinality 'k'.

Usage

```
oscar.bs.k(bs)
```

Arguments

bs Bootstrapped list from oscar.bs

Value

Reformatted data.frame

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_bs <- oscar.bs(fit, bootstrap = 20, seed = 123)
  ll <- oscar.bs.k(fit_bs)
  head(ll)
  tail(ll)
}
```

oscar.bs.plot	<i>Bootstrap heatmap plot for oscar models</i>
---------------	--

Description

This function neatly plots a colourized proportion of variables chosen as a function of cardinalities over a multitude of bootstrap runs. This helps model diagnostics in assessing variable importance.

Usage

```
oscar.bs.plot(
  fit,
  bs,
  kmax,
  cex.axis = 0.6,
  palet = colorRampPalette(c("orange", "red", "black", "blue", "cyan"))(dim(bs)[3]),
  nbins = dim(bs)[3],
  Colv = NA,
  Rowv = NA,
  ...
)
```

Arguments

<code>fit</code>	Fitted oscar S4-class object
<code>bs</code>	Bootstrapped 3-dimensional array for an oscar object as produced by <code>oscar.bs</code>
<code>kmax</code>	Maximum cardinality 'k'
<code>cex.axis</code>	Axis magnification
<code>palet</code>	Colour palette
<code>nbins</code>	Number of bins (typically ought to be same as number of colours in the palette)
<code>Colv</code>	Column re-ordering indices or a readily built dendrogram
<code>Rowv</code>	Row re-ordering indices or a readily built dendrogram
<code>...</code>	Additional parameters passed on to the <code>hamlet::hmap</code> function

Details

Further heatmap parameters available from `?hmap`

Value

This is a plotting function that does not return anything, but instead draws on a new graphics device.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_bs <- oscar.bs(fit, bootstrap = 20, seed = 123)
  oscar.bs.plot(fit, fit_bs)
}
```

oscar.bs.visu	<i>Visualize bootstrapping of a fit oscar object</i>
---------------	--

Description

This function visualizes bootstrapped model coefficients over multiple bootstrap runs as lines in a graph

Usage

```
oscar.bs.visu(bs, intercept = FALSE, add = FALSE)
```

Arguments

bs	Bootstrapped 3-dimensional array for an oscar object as produced by oscar.bs
intercept	Whether model intercept should be plotted also as a coefficient, Default: FALSE
add	Should plot be added on top of an existing plot device

Value

This is a plotting function that does not return anything, but instead draws on an existing or a new graphics device.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_bs <- oscar.bs(fit, bootstrap = 20, seed = 123)
  oscar.bs.visu(fit_bs)
}
```

oscar.control	<i>Control OSCAR optimizer parameters</i>
---------------	---

Description

Fine-tuning the parameters available for the DBDC and LMBM optimizers. See oscar documentation for the optimization algorithms for further details.

Usage

```

oscar.control(
  x,
  family,
  start = 2,
  in_mrounds = 5000,
  in_mit = 5000,
  in_mrounds_esc = 5000,
  in_b1,
  in_b2 = 3,
  in_b,
  in_m = 0.01,
  in_m_clarke = 0.01,
  in_c = 0.1,
  in_r_dec,
  in_r_inc = 10^5,
  in_eps1 = 5 * 10^(-5),
  in_eps,
  in_crit_tol = 10^(-5),
  na = 4,
  mcu = 7,
  mcinit = 7,
  tolf = 10^(-5),
  tolf2 = 10^4,
  tolg = 10^(-5),
  tolg2 = tolg,
  eta = 0.5,
  epsL = 0.125
)

```

Arguments

x	Input data matrix 'x'; will be used for calculating various control parameter defaults.
family	Model family; should be one of 'cox', 'logistic', or 'gaussian'/'mse'
start	Starting point generation method, see vignettes for details; should be an integer between range,range, Default: 2
in_mrounds	DBDC: The maximum number of rounds in one main iteration, Default: 5000
in_mit	DBDC: The maximum number of main iterations, Default: 5000
in_mrounds_esc	DBDC: The maximum number of rounds in escape procedure, Default: 5000
in_b1	DBDC: The size of bundle B1, Default: min(n_feat+5,1000)
in_b2	DBDC: The size of bundle B2, Default: 3
in_b	DBDC: Bundle B in escape procedure, Default: 2*n_feat
in_m	DBDC: The descent parameter in main iteration, Default: 0.01
in_m_clarke	DBDC: The descent parameter in escape procedure, Default: 0.01

<code>in_c</code>	DBDC: The extra decrease parameter in main iteration, Default: 0.1
<code>in_r_dec</code>	DBDC: The decrease parameter in main iteration, Default: 0.75, 0.99, or larger depending on <code>n_obs</code> (thresholds 10, 300, and above)
<code>in_r_inc</code>	DBDC: The increase parameter in main iteration, Default: 10^5
<code>in_eps1</code>	DBDC: The enlargement parameter, Default: $5 \cdot 10^{-5}$
<code>in_eps</code>	DBDC: The stopping tolerance (proximity measure), Default: 10^{-6} if number of features is ≤ 50 , otherwise 10^{-5}
<code>in_crit_tol</code>	DBDC: The stopping tolerance (criticality tolerance), Default: 10^{-5}
<code>na</code>	LMBM: Size of the bundle, Default: 4
<code>mcu</code>	LMBM: Upper limit for maximum number of stored corrections, Default: 7
<code>mcinit</code>	LMBM: Initial maximum number of stored corrections, Default: 7
<code>tolf</code>	LMBM: Tolerance for change of function values, Default: 10^{-5}
<code>tolf2</code>	LMBM: Second tolerance for change of function values, Default: 10^{-4}
<code>tolg</code>	LMBM: Tolerance for the first termination criterion, Default: 10^{-5}
<code>tolg2</code>	LMBM: Tolerance for the second termination criterion, Default: same as <code>'tolg'</code>
<code>eta</code>	LMBM: Distance measure parameter (>0), Default: 0.5
<code>epsL</code>	LMBM: Line search parameter ($0 < \text{epsL} < 0.25$), Default: 0.125

Details

This function sanity checks and provides reasonable DBDC ('Double Bundle method for non-smooth DC optimization' as described in Joki et al. (2018) <doi:10.1137/16M1115733>) and LMBM ('Limited Memory Bundle Method for large-scale nonsmooth optimization' as presented in Haarala et al. (2004) <doi:10.1080/10556780410001689225>) optimization tuning parameters. User may override custom values, though sanity checks will prevent unreasonable values and replace them. The returned list of parameters can be provided for the `'control'` parameter when fitting oscar-objects.

Value

A list of sanity checked parameter values for the OSCAR optimizers.

Examples

```
if(interactive()){
  oscar.control() # Return a list of default parameters
}
```

oscar.cost.after	<i>Return total cost of model fits if the cost is not included in the oscar object</i>
------------------	--

Description

If at least one measurement from a kit is included in the model, the kit cost is added.

Usage

```
oscar.cost.after(object)
```

Arguments

object	Fit oscar S4-object
--------	---------------------

Value

A vector for numeric values of total kit costs at different cardinalities.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  oscar.cost.after(fit)
}
```

oscar.cv	<i>Cross-validation for oscar-fitted model objects over k-range</i>
----------	---

Description

Create a cross-validation matrix with the chosen goodness metric with n-folds. Based on the goodness metric, one ought to pick optimal cardinality (parameter 'k').

Usage

```
oscar.cv(
  fit,
  fold = 10,
  seed = NULL,
  strata = rep(1, times = nrow(fit@x)),
  verb = 0,
  ...
)
```

Arguments

<code>fit</code>	oscar-model object
<code>fold</code>	Number of cross-validation folds, Default: 10
<code>seed</code>	Random seed for reproducibility with NULL indicating that it is not set, Default: NULL
<code>strata</code>	Should stratified cross-validation be used; separate values indicate balanced strata. Default: Unit vector, which will treat all observations equally.
<code>verb</code>	Level of verbosity with higher integer giving more information, Default: 0
<code>...</code>	Additional parameters passed to oscar-function

Details

A k-fold cross-validation is run by mimicking the parameters contained in the original oscar S4-object. This requires the original data at slots `@x` and `@y`.

Value

A matrix with goodness of fit over folds and k-values

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_cv <- oscar.cv(fit, fold=10, seed=123)
  fit_cv
}
```

oscar.cv.visu

Visualize cross-validation as a function of k

Description

This function plots the model performance as a function of cardinality for k-fold cross-validation. Performance metric depends on user choice and model family (i.e. lower MSE is good, higher C-index is good).

Usage

```
oscar.cv.visu(
  cv,
  add = FALSE,
  main = "OSCAR cross-validation",
  xlab = "Cardinality 'k'",
  ylab = "CV performance",
  ...
)
```

Arguments

cv	Matrix produced by oscar.cv; rows are cv-folds, cols are k-values
add	Should plot be added on top of an existing plot device
main	Main title
xlab	X-axis label
ylab	Y-axis label
...	Additional parameters passed on top the CV points

Value

This is a plotting function that does not return anything, but instead draws on an existing or a new graphics device.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_cv <- oscar.cv(fit, fold = 10, seed = 123)
  oscar.cv.visu(fit_cv)
}
```

oscar.pareto	<i>Retrieve a set of pareto-optimal points for an oscar-model based on model goodness-of-fit or cross-validation</i>
--------------	--

Description

This function retrieves the set of pareto optimal points for an oscar model fit in n-proportional time as cardinality axis is readily sorted. It is advisable to optimize model generalization (via cross-validation) rather than mere goodness-of-fit.

Usage

```
oscar.pareto(fit, cv, xval = "cost", weak = FALSE, summarize = mean)
```

Arguments

fit	Fit oscar S4-object
cv	A cross-validation matrix as produced by oscar.cv; if CV is not provided, then goodness-of-fit from fit object itself is used rather than cross-validation generalization metric
xval	The x-axis to construct pareto front based on; by default 'cost' vector for features/kits, can also be 'cardinality'/'k'
weak	If weak pareto-optimality is allowed; by default FALSE.
summarize	Function that summarizes over cross-validation folds; by default, this is the mean over the k-folds.

Value

A data.frame containing points and indices at which pareto optimal points exist

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_cv <- oscar.cv(fit, fold=10)
  oscar.pareto(fit, cv=fit_cv)
}
```

oscar.pareto.visu	<i>Visualize oscar model pareto front</i>
-------------------	---

Description

Visualization function for showing the pareto front for cardinality 'k' and model goodness metric, either from goodness-of-fit or from cross-validation

Usage

```
oscar.pareto.visu(
  fit,
  cv,
  xval = "cost",
  weak = FALSE,
  summarize = mean,
  add = FALSE,
  ...
)
```

Arguments

fit	Fit oscar S4-object
cv	A cross-validation matrix as produced by oscar.cv; if CV is not provided, then goodness-of-fit from fit object itself is used rather than cross-validation generalization metric
xval	The x-axis to construct pareto front based on; by default 'cost' vector for features/kits, can also be 'cardinality'/'k'
weak	If weak pareto-optimality is allowed; by default FALSE.
summarize	Function that summarizes over cross-validation folds; by default, this is the mean over the k-folds.
add	If the fit should be added on top of an existing plot; in that case leaving out labels etc. By default new plot is called.
...	Additional parameters provided for the plotting functions

Value

This is a plotting function that does not return anything, but instead draws on an existing or a new graphics device.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  fit_cv <- oscar.cv(fit, fold = 10, seed = 123)
  opar <- par(mfrow=c(1,2))
  oscar.pareto.visu(fit=fit) # Model goodness-of-fit
  oscar.pareto.visu(fit=fit, cv=fit_cv) # Model cross-validation performance
  par(opar)
}
```

oscar.sparsify

Create a sparse matrix representation of betas as a function of k

Description

Variable estimates (rows) as a function of cardinality (k, columns). Since a model can drop out variables in favor of two better ones as k increases, this sparse representation helps visualize which variables are included at what cardinality.

Usage

```
oscar.sparsify(fit, kmax = fit@kmax)
```

Arguments

fit	oscar-model object
kmax	Create matrix until kmax-value; by default same as for fit object, but for high dimensional tasks one may wish to reduce this

Details

Uses sparseMatrix-class from Matrix-package

Value

A sparse matrix of variables (rows) as a function of cardinality k (columns), where elements are the beta estimates.

Examples

```
if(interactive()){
  data(ex)
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')
  oscar.sparsify(fit, kmax=5)
}
```

oscar.visu	<i>Target function value and total kit cost as a function of number of kits included</i>
------------	--

Description

Plot oscar S4-object goodness-of-fit, kit costs, and similar performance metrics.

Usage

```
oscar.visu(
  fit,
  y = c("target", "cost", "goodness", "cv", "AIC"),
  cols = c("red", "blue"),
  legend = "top",
  mtexts = TRUE,
  add = FALSE,
  main = ""
)
```

Arguments

fit	Fitted oscar S4-class object
y	Plotted y-axes supporting two simultaneous axes with different scales, Default: c("target", "cost", "goodness", "cv")
cols	Colours for drawn lines, Default: c("red", "blue")
legend	Location of legend or omission of legend with NA, Default: 'top'
mtexts	Outer margin texts
add	Should plot be added into an existing frame / plot
main	Main title

Value

This is a plotting function that does not return anything, but instead draws on an existing or a new graphics device.

Examples

```
if(interactive()){  
  data(ex)  
  fit <- oscar(x=ex_X, y=ex_Y, k=ex_K, w=ex_c, family='cox')  
  oscar.visu(fit, y=c("target", "cost"))  
}
```

show,oscar-method	<i>Showing oscar-objects</i>
-------------------	------------------------------

Description

Showing oscar-objects

Usage

```
## S4 method for signature 'oscar'  
show(object)
```

Arguments

object	Fit oscar S4-object
--------	---------------------

Value

Outputs raw text describing key characteristics of the oscar-object

Index

`coef`, `oscar-method`, 3
`cost`, 4
`cost`, `oscar-method (cost)`, 4
`cost`, `oscar-methods (cost)`, 4

`ex_c` (Example data), 5
`ex_K` (Example data), 5
`ex_X` (Example data), 5
`ex_Y` (Example data), 5
Example data, 5

`feat`, 5
`feat`, `oscar-method (feat)`, 5
`feat`, `oscar-methods (feat)`, 5

`kits`, 6
`kits`, `oscar-method (kits)`, 6
`kits`, `oscar-methods (kits)`, 6

`oscar`, 6
`oscar-class`, 8
`oscar-package`, 3
`oscar.binarize`, 8
`oscar.binplot`, 8, 9
`oscar.bs`, 8, 10
`oscar.bs.boxplot`, 11
`oscar.bs.k`, 12
`oscar.bs.plot`, 12
`oscar.bs.visu`, 8, 14
`oscar.control`, 14
`oscar.cost.after`, 17
`oscar.cv`, 8, 17
`oscar.cv.visu`, 8, 18
`oscar.pareto`, 8, 19
`oscar.pareto.visu`, 8, 20
`oscar.sparsify`, 21
`oscar.visu`, 8, 22

`plot`, `oscar-method (coef, oscar-method)`, 3
`predict`, `oscar-method`
 (`coef`, `oscar-method`), 3

`show`, `oscar-method`, 23