# Package: ICEbox (via r-universe)

October 29, 2026

**Type** Package

**Title** Individual Conditional Expectation Plot Toolbox

**Version** 1.2

**Date** 2026-01-11

**Author** Alex Goldstein [aut], Adam Kapelner [aut, cre] (ORCID: <https://orcid.org/0000-0001-5985-6792>), Justin Bleich [aut]

**Maintainer** Adam Kapelner <kapelner@qc.cuny.edu>

**Description** Implements Individual Conditional Expectation (ICE) plots, a tool for visualizing the model estimated by any supervised learning algorithm. ICE plots refine Friedman's partial dependence plot by graphing the functional relationship between the predicted response and a covariate of interest for individual observations. Specifically, ICE plots highlight the variation in the fitted values across the range of a covariate of interest, suggesting where and to what extent they may exist.

**License** GPL-2 | GPL-3

**URL** https://github.com/kapelner/ICEbox

**BugReports** https://github.com/kapelner/ICEbox/issues

**Imports** ggplot2, checkmate, data.table, Rcpp

**LinkingTo** Rcpp

**Suggests** randomForest, MASS, testthat, rpart

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**NeedsCompilation** yes

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

**Date/Publication** 2026-01-12 06:00:02 UTC

**RemoteUrl** https://github.com/cran/ICEbox

**RemoteRef** HEAD

**RemoteSha** b805366b66596c66987b49bfe09e5e6b1c9ca057

# Contents

additivityLineup . . . 2
backfitter . . . 3
clusterICE . . . 4
colSds_cpp . . . 6
derivative_cpp . . . 6
dice . . . 7
ice . . . 9
melt_ice_curves_cpp . . . 12
plot.dice . . . 12
plot.ice . . . 14
print.dice . . . 17
print.ice . . . 18
rowCenter_cpp . . . 18
sg_smooth_cpp . . . 19
summary.dice . . . 19
summary.ice . . . 20
transform_ice_curves_cpp . . . 20
WhiteWine . . . 21

**Index** **22**

---

`additivityLineup` *Lineup plot for additivity*

---

## Description

This function creates a lineup plot to assess the additivity of a predictor's effect. It uses a nonparametric bootstrap approach to generate null plots.

## Usage

```
additivityLineup(
  backfit_obj,
  fitMethod,
  realICE,
  figs = 10,
  colorvecfcn,
  usecolorvecfcn_inreal = FALSE,
  null_predictfcn,
  ...
)
```

### Arguments

| | |
|---|---|
| backfit_obj | An object of class backfitter. |
| fitMethod | A function that accepts X and y and returns a fitted model. |
| realICE | The ice object for the real data. |
| figs | The total number of plots in the lineup (including the real one). Default is 10. |
| colorvecfcn | Optional function to generate a color vector for the curves. |
| usecolorvecfcn_inreal | If TRUE, use colorvecfcn for the real plot. |
| null_predictfcn | Optional prediction function for the null models. |
| ... | Additional arguments passed to plot.ice. |

### Value

An object of class additivityLineup (invisibly).

---

## backfitter *Backfitting for Additive Models*

---

### Description

Fits a model of the form $\hat{f}(x) = \hat{g}_1(x_S) + \hat{g}_2(x_C)$ using backfitting.

### Usage

```
backfitter(
  X,
  y,
  predictor,
  fitMethod,
  predictfcn,
  eps = 0.01,
  iter.max = 10,
  verbose = TRUE,
  ...
)
```

### Arguments

| | |
|---|---|
| X | The design matrix. |
| y | The response vector. |
| predictor | The name or index of the predictor of interest ($x_S$). |
| fitMethod | A function that accepts X and y and returns a fitted model. |
| predictfcn | A function that accepts object and newdata and returns predictions. |

| | |
|---|---|
| eps | Convergence threshold. |
| iter.max | Maximum number of iterations. |
| verbose | If TRUE, prints progress messages. |
| ... | Additional arguments passed to fitMethod. |

## Value

An object of class backfitter.

---

clusterICE *Clustering of ICE and d-ICE curves by kmeans.*

---

## Description

Clustering if ICE and d-ICE curves by kmeans. All curves are centered to have mean 0 and then kmeans is applied to the curves with the specified number of clusters.

## Usage

```
clusterICE(
  ice_obj,
  nClusters,
  plot = TRUE,
  plot_margin = 0.05,
  colorvec,
  plot_pdp = FALSE,
  x_quantile = FALSE,
  avg_lwd = 3,
  centered = FALSE,
  plot_legend = FALSE,
  main = NULL,
  num_cores = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| ice_obj | Object of class ice or dice to cluster. |
| nClusters | Number of clusters to find. |
| plot | If TRUE, plots the clusters. |
| plot_margin | Extra margin to pass to ylim as a fraction of the range of cluster centers. |
| colorvec | Optional vector of colors to use for each cluster. |
| plot_pdp | If TRUE, the PDP (ice object) or d-PDP (dice object) is plotted with a dotted black line and highlighted in yellow. |

| | |
|---|---|
| x_quantile | If TRUE, the plot is drawn with the x-axis taken to be quantile(gridpts). If FALSE, the predictor's original scale is used. |
| avg_lwd | Average line width to use when plotting the cluster means. Line width is proportional to the cluster's size. |
| centered | If TRUE, all cluster means are shifted to be to be 0 at the minimum value of the predictor. If FALSE, the original cluster means are used. |
| plot_legend | If TRUE a legend mapping line colors to the proportion of the data in each cluster is added to the plot. |
| main | Optional title for the plot. |
| num_cores | Integer number of cores to use for parallel operations. Default is 1. |
| ... | Additional arguments for plotting. |

## Value

A list with the following elements:

| | |
|---|---|
| cl | The output of the kmeans call (a list of class kmeans). |
| plot | The ggplot object used for plotting (if plot = TRUE). |

## See Also

ice, dice

## Examples

```
## Not run:
require(ICEbox)
require(randomForest)
require(MASS) #has Boston Housing data, Pima

data(Boston) #Boston Housing data
X = Boston
y = X$medv
X$medv = NULL

## build a RF:
bh_rf = randomForest(X, y)

## Create an 'ice' object for the predictor "age":
bh.ice = ice(object = bh_rf, X = X, y = y, predictor = "age",
            frac_to_build = .1)

## cluster the curves into 2 groups.
clusterICE(bh.ice, nClusters = 2, plot_legend = TRUE)

## cluster the curves into 3 groups, start all at 0.
clusterICE(bh.ice, nClusters = 3, plot_legend = TRUE, center = TRUE)

## End(Not run)
```

## colSds_cpp *Efficient Column Standard Deviations*

### Description

Efficient Column Standard Deviations

### Usage

```
colSds_cpp(x, n_cores = 1L)
```

### Arguments

| | |
|---|---|
| x | Numeric Matrix |
| n_cores | Number of cores to use |

## derivative_cpp *Efficient Numerical Derivative for Matrix (Row-wise)*

### Description

Computes the first derivative using centered differences, mirroring sfsmisc::D1tr.

### Usage

```
derivative_cpp(x, gridpts, n_cores = 1L)
```

### Arguments

| | |
|---|---|
| x | Numeric Matrix (smoothed values) |
| gridpts | Grid points corresponding to columns of x |
| n_cores | Number of cores to use |

dice *Creates an object of class* dice.

## Description

Estimates the partial derivative function for each curve in an ice object. See Goldstein et al (2013) for further details.

## Usage

```
dice(
  ice_obj,
  DerivEstimator = NULL,
  use_supsmu = FALSE,
  verbose = TRUE,
  num_cores = 1,
  sg_poly_order = 2,
  sg_window_size = NULL
)
```

## Arguments

| | |
|---|---|
| ice_obj | Object of class ice. This function generates partial derivative estimates for each row in ice_obj$ice_curves. |
| DerivEstimator | Optional function with a single argument y. Returns the estimated partial derivative of a function sampled at the points (ice_obj$gridpts,y). If NULL, the default uses a Savitzky-Golay filter to estimate the first derivative. |
| use_supsmu | If TRUE, uses the old supsmu based derivative estimation logic. This is much slower than the default Savitzky-Golay filter. |
| verbose | If TRUE, prints messages about the procedure's progress. |
| num_cores | Integer number of cores to use for parallel derivative estimation. Defaults to 1. |
| sg_poly_order | Polynomial order for Savitzky-Golay filter. Default is 2. |
| sg_window_size | Window size for Savitzky-Golay filter. Default is 30% of the grid. |

## Value

A list of class dice with the following elements. Most are passed directly through from ice_object and exist to enable various plotting facilities.

| | |
|---|---|
| d_ice_curves | Matrix of dimension nrow(Xice) by length(gridpts). Each row corresponds to an observation's d-ICE curve, estimated at the values of predictor in gridpts. |
| xj | The actual values of predictor observed in the data in the order of Xice. |
| actual_deriv | Vector of length nrow(Xice) containing the estimated partial derivatives at the value of the predictor actually found in Xice. |

| | |
|---|---|
| sd_deriv | Vector of length length(gridpts) with the cross-observation sd of partial derivative estimates. For instance sd_deriv[1] equals sd(d_ice_curves[,1]). |
| logodds | Passed from ice_object. If TRUE, d_ice_curves are estimated derivatives of the centered log-odds. |
| gridpts | Passed from ice_object. |
| predictor | Passed from ice_object. |
| xlab | Passed from ice_object. |
| nominal_axis | Passed from ice_object. |
| range_y | Passed from ice_object. |
| Xice | Passed from ice_object. |
| dpdp | The estimated partial derivative of the PDP. |

## References

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E., Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. (2014) Journal of Computational and Graphical Statistics, in press

## See Also

ice, dice

## Examples

```
## Not run:
# same examples as for 'ice', but now create a derivative estimate as well.
require(ICEbox)
require(randomForest)
require(MASS) #has Boston Housing data, Pima

########  regression example
data(Boston) #Boston Housing data
X = Boston
y = X$medv
X$medv = NULL

## build a RF:
bhd_rf_mod = randomForest(X, y)

## Create an 'ice' object for the predictor "age":
bhd.ice = ice(object = bhd_rf_mod, X = X, y = y, predictor = "age", frac_to_build = .1)

# make a dice object:
bhd.dice = dice(bhd.ice)

#### classification example
data(Pima.te)  #Pima Indians diabetes classification
y = Pima.te$type
```

```
X = Pima.te
X$type = NULL

## build a RF:
pima_rf = randomForest(x = X, y = y)

## Create an 'ice' object for the predictor "skin":
# For classification we plot the centered log-odds. If we pass a predict
# function that returns fitted probabilities, setting logodds = TRUE instructs
# the function to set each ice curve to the centered log-odds of the fitted
# probability.
pima.ice = ice(object = pima_rf, X = X, predictor = "skin", logodds = TRUE,
                    predictfcn = function(object, newdata){
                         predict(object, newdata, type = "prob")[, 2]
                    }
               )

# make a dice object:
pima.dice = dice(pima.ice)

## End(Not run)
```

---

ice *Creates an object of class* ice.

---

## Description

Creates an ice object with individual conditional expectation curves for the passed model object, X matrix, predictor, and response. See Goldstein et al (2013) for further details.

## Usage

```
ice(
  object,
  X,
  y,
  predictor,
  predictfcn,
  verbose = TRUE,
  frac_to_build = 1,
  indices_to_build = NULL,
  num_grid_pts,
  logodds = FALSE,
  probit = FALSE,
  num_cores = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| object | The fitted model to estimate ICE curves for. |
| X | The design matrix we wish to estimate ICE curves for. Rows are observations, columns are predictors. Typically this is taken to be object's training data, but this is not strictly necessary. |
| y | Optional vector of the response values object was trained on. It is used to compute y-axis ranges that are useful for plotting. If not passed, the range of predicted values is used and a warning is printed. |
| predictor | The column number or variable name in X of the predictor of interest, ($x_S = X[, j]$). |
| predictfcn | Optional function that accepts two arguments, object and newdata, and returns an N vector of object's predicted response for data newdata. If this argument is not passed, the procedure attempts to find a generic predict function corresponding to class(object). |
| verbose | If TRUE, prints messages about the procedure's progress. |
| frac_to_build | Number between 0 and 1, with 1 as default. For large X matrices or fitted models that are slow to make predictions, specifying frac_to_build less than 1 will choose a subset of the observations to build curves for. The subset is chosen such that the remaining observations' values of predictor are evenly spaced throughout the quantiles of the full X[,predictor] vector. |
| indices_to_build | Vector of indices, $\subset \{1, \ldots, nrow(X)\}$ specifying which observations to build ICE curves for. As this is an alternative to setting frac_to_build, both cannot be specified. |
| num_grid_pts | Optional number of values in the range of predictor at which to estimate each curve. If missing, the curves are estimated at each unique value of predictor in the X observations we estimate ICE curves for. |
| logodds | If TRUE, for classification creates PDPs by plotting the centered log-odds implied by the fitted probabilities. We assume that the generic or passed predict function returns probabilities, and so the flag tells us to transform these to centered logits after the predictions are generated. Note: probit cannot be TRUE. |
| probit | If TRUE, for classification creates PDPs by plotting the probit implied by the fitted probabilities. We assume that the generic or passed predict function returns probabilities, and so the flag tells us to transform these to probits after the predictions are generated. Note: logodds cannot be TRUE. |
| num_cores | Integer number of cores to use for parallel prediction. Defaults to 1. |
| ... | Other arguments to be passed to object's generic predict function. |

## Value

A list of class ice with the following elements:

| | |
|---|---|
| gridpts | Sorted values of predictor at which each curve is estimated. Duplicates are removed – by definition, elements of gridpts are unique. |

`ice_curves` Matrix of dimension `nrow(X)` by `length(gridpts)`. Each row corresponds to an observation's ICE curve, estimated at the values of `predictor` in `gridpts`.

`xj` The actual values of `predictor` observed in the data in the order of `Xice`.

`actual_prediction` Vector of length `nrow(X)` containing the model's predictions at the actual value of the predictors in the order of `Xice`.

`xlab` String with the predictor name corresponding to `predictor`. If `predictor` is a column number, `xlab` is set to `colnames(X)[, predictor]`.

`nominal_axis` If `TRUE`, `length(gridpts)` is 5 or fewer; otherwise `FALSE`. When `TRUE` the `plot` function treats the x-axis as if x is nominal.

`range_y` If `y` was passed, the range of the response. Otherwise it defaults to be `max(ice_curves) - min(ice_curves)` and a message is printed to the console.

`sd_y` If `y` was passed, the standard deviation of the response. Otherwise it is defaults to `sd(actual_prediction)` and a message is printed to the console.

`Xice` A matrix containing the subset of `X` for which ICE curves are estimated. Observations are ordered to be increasing in `predictor`. This ordering is the same one as in `ice_curves`, `xj` and `actual_prediction`, meaning for all these objects the `i`-th element refers to the same observation in `X`.

`pdp` A vector of size `length(gridpts)` which is a numerical approximation to the partial dependence function (PDP) corresponding to the estimated ICE curves. See Goldstein et al (2013) for a discussion of how the PDP is a form of post-processing. See Friedman (2001) for a description of PDPs.

`predictor` Same as the argument, see argument description.

`logodds` Same as the argument, see argument description.

`indices_to_build` Same as the argument, see argument description.

`frac_to_build` Same as the argument, see argument description.

`predictfcn` Same as the argument, see argument description.

## References

Jerome Friedman. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 29(5): 1189-1232, 2001.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E., Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. (2014) Journal of Computational and Graphical Statistics, in press

## See Also

`plot.ice`, `print.ice`, `summary.ice`

## Examples

```
## Not run:
require(ICEbox)
require(randomForest)
require(MASS) #has Boston Housing data, Pima

########  regression example
data(Boston) #Boston Housing data
X = Boston
y = X$medv
X$medv = NULL

## build a RF:
bhd_rf_mod = randomForest(X, y)

## Create an 'ice' object for the predictor "age":
bhd.ice = ice(object = bhd_rf_mod, X = X, y = y, predictor = "age", frac_to_build = .1)

## End(Not run)
```

---

`melt_ice_curves_cpp` *Melt Matrix to Long Format Vector*

---

## Description

Efficiently converts a matrix to a long-format vector (row-major order) for plotting.

## Usage

```
melt_ice_curves_cpp(x, n_cores = 1L)
```

## Arguments

| | |
|---|---|
| `x` | Numeric Matrix |
| `n_cores` | Number of cores to use |

---

`plot.dice` *Create a plot of a* `dice` *object.*

---

## Description

Plotting of `dice` objects.

## Usage

```
## S3 method for class 'dice'
plot(
  x,
  plot_margin = 0.05,
  frac_to_plot = 1,
  plot_sd = TRUE,
  plot_orig_pts_deriv = TRUE,
  pts_preds_size = 1.5,
  colorvec,
  color_by = NULL,
  x_quantile = TRUE,
  plot_dpdp = TRUE,
  rug_quantile = seq(from = 0, to = 1, by = 0.1),
  verbose = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| x | Object of class dice to plot. |
| plot_margin | Extra margin to pass to ylim as a fraction of the range of x$d_ice_curves. |
| frac_to_plot | If frac_to_plot is less than 1, randomly plot frac_to_plot fraction of the curves in x$d_ice_curves. |
| plot_sd | If TRUE, plot the cross-observation sd of partial derivatives below the derivative plots. |
| plot_orig_pts_deriv | If TRUE, marks each curve at the location of the derivative estimate at the location of predictor actually occurring in the data. If FALSE no mark is drawn. |
| pts_preds_size | Size of points to make if plot_orig_pts_deriv is TRUE. |
| colorvec | Optional vector of colors to use for each curve. |
| color_by | Optional variable name (or column number) in Xice to color curves by. If the color_by variable has 10 or fewer unique values, a discrete set of colors is used for each value and a legend is printed and returned. If there are more values, curves are colored from light to dark corresponding to low to high values of the variable specified by color_by. |
| x_quantile | If TRUE, the plot is drawn with the x-axis taken to be quantile(gridpts). If FALSE, the predictor's original scale is used. |
| plot_dpdp | If TRUE, the estimated derivative of the PDP is plotted and highlighted in yellow. |
| rug_quantile | If not null, tick marks are drawn on the x-axis corresponding to the vector of quantiles specified by this parameter. Forced to NULL when x_quantile is set to TRUE. |
| verbose | If TRUE, prints the color legend to the console. |
| ... | Additional plotting arguments. |

## Value

A list with the following elements.

`plot_points_indices`
: Row numbers of `Xice` of those observations presented in the plot.

`legend_text`
: If the `color_by` argument was used, a legend describing the map between the `color_by` predictor and curve colors.

`plot`
: The ggplot object used for plotting.

## See Also

`dice`

## Examples

```
## Not run:
require(ICEbox)
require(randomForest)
require(MASS) #has Boston Housing data, Pima

data(Boston) #Boston Housing data
X = Boston
y = X$medv
X$medv = NULL

## build a RF:
bhd_rf_mod = randomForest(X, y)

## Create an 'ice' object for the predictor "age":
bhd.ice = ice(object = bhd_rf_mod, X = X, y = y, predictor = "age", frac_to_build = .1)

# estimate derivatives, then plot.
bhd.dice = dice(bhd.ice)
plot(bhd.dice)

## End(Not run)
```

---

# plot.ice *Plotting of* ice *objects.*

---

## Description

Plotting of `ice` objects.

## Usage

```
## S3 method for class 'ice'
plot(
  x,
  plot_margin = 0.05,
  frac_to_plot = 1,
  plot_points_indices = NULL,
  plot_orig_pts_preds = TRUE,
  pts_preds_size = 1.5,
  colorvec,
  color_by = NULL,
  x_quantile = TRUE,
  plot_pdp = TRUE,
  centered = FALSE,
  prop_range_y = TRUE,
  rug_quantile = seq(from = 0, to = 1, by = 0.1),
  centered_percentile = 0,
  point_labels = NULL,
  point_labels_size = NULL,
  prop_type = "sd",
  verbose = TRUE,
  num_cores = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| x | Object of class ice to plot. |
| plot_margin | Extra margin to pass to ylim as a fraction of the range of x$ice_curves. |
| frac_to_plot | If frac_to_plot is less than 1, randomly plot frac_to_plot fraction of the curves in x$ice_curves. |
| plot_points_indices | If not NULL, this plots only the indices of interest. If not NULL, frac_to_plot must be 1 otherwise an error is thrown. Default is NULL. |
| plot_orig_pts_preds | If TRUE, marks each curve at the location of the observation's actual fitted value. If FALSE, no mark is drawn. |
| pts_preds_size | Size of points to make if plot_origin_pts_preds is TRUE. |
| colorvec | Optional vector of colors to use for each curve. |
| color_by | Optional variable name in Xice, column number in Xice, or data vector of the correct length to color curves by. If the color_by variable has 10 or fewer unique values, a discrete set of colors is used for each value and a legend is printed and returned. If there are more values, curves are colored from light to dark corresponding to low to high values of the variable specified by color_by. |
| x_quantile | If TRUE, the plot is drawn with the x-axis taken to be quantile(gridpts). If FALSE, the predictor's original scale is used. |

plot_pdp — If TRUE, the PDP is plotted and highlighted in yellow.

centered — If TRUE, all curves are re-centered to be 0 at the quantile given by centered_percentile. See Goldstein et al (2013) for details and examples. If FALSE, the original ice_curves are plotted.

prop_range_y — When TRUE and centered=TRUE as well, the range of the right vertical axis displays the centered values as a fraction of the sd of the fitted values on actual observations if prop_type is missing or set to "sd". If prop_type is set to "range", the right axis displays the centered values as a fraction of the range of the fitted values over the actual observations.

rug_quantile — If not NULL, tick marks are drawn on the x-axis corresponding to the vector of quantiles specified by this parameter. Forced to NULL when x_quantile is set to TRUE.

centered_percentile — The percentile of predictor for which all ice_curves are "pinched together" and set to be 0. Default is 0.

point_labels — If not NULL, labels to plot next to each point. Default is NULL.

point_labels_size — If not NULL, size of labels to plot next to each point. Default is NULL which means it's the size of pts_preds_size.

prop_type — Scaling factor for the right vertical axis in centered plots if prop_range_y is TRUE. Can be one of "sd" (default) or "range". Ignored if centered and prop_range_y are not both TRUE.

verbose — If TRUE, prints the color legend to the console.

num_cores — Used for parallel plotting speedup. Default is 1.

... — Other arguments to be passed to the plot function.

## Value

A list with the following elements.

plot_points_indices — Row numbers of Xice of those observations presented in the plot.

legend_text — If the color_by argument was used, a legend describing the map between the color_by predictor and curve colors.

## See Also

ice

## Examples

```
## Not run:
require(ICEbox)
require(randomForest)
require(MASS) #has Boston Housing data, Pima

data(Boston) #Boston Housing data
```

```
X = Boston
y = X$medv
X$medv = NULL

## build a RF:
bhd_rf_mod = randomForest(X, y)

## Create an 'ice' object for the predictor "age":
bhd.ice = ice(object = bhd_rf_mod, X = X, y = y, predictor = "age",
            frac_to_build = .1)

## plot
plot(bhd.ice, x_quantile = TRUE, plot_pdp = TRUE, frac_to_plot = 1)

## centered plot
plot(bhd.ice, x_quantile = TRUE, plot_pdp = TRUE, frac_to_plot = 1,
 centered = TRUE)

## color the curves by high and low values of 'rm'.
# First create an indicator variable which is 1 if the number of
# rooms is greater than the median:
median_rm = median(X$rm)
bhd.ice$Xice$I_rm = ifelse(bhd.ice$Xice$rm > median_rm, 1, 0)

plot(bhd.ice, frac_to_plot = 1, centered = TRUE, prop_range_y = TRUE,
            x_quantile = T, plot_orig_pts_preds = T, color_by = "I_rm")
bhd.ice = ice(object = bhd_rf_mod, X = X, y = y, predictor = "age",
            frac_to_build = 1)
plot(bhd.ice, frac_to_plot = 1, centered = TRUE, prop_range_y = TRUE,
            x_quantile = T, plot_orig_pts_preds = T, color_by = y)

## End(Not run)
```

---

`print.dice` *Print method for* `dice` *objects.*

---

## Description

Prints a summary of a `dice` object.

## Usage

```
## S3 method for class 'dice'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object of class `dice`. |
| `...` | Ignored for now. |

## print.ice *Print method for* ice *objects.*

### Description

Prints a summary of an ice object.

### Usage

```
## S3 method for class 'ice'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object of class ice. |
| ... | Ignored for now. |

## rowCenter_cpp *Row-wise Centering*

### Description

Centers each row of a matrix by subtracting the row mean.

### Usage

```
rowCenter_cpp(x, n_cores = 1L)
```

### Arguments

| | |
|---|---|
| x | Numeric Matrix |
| n_cores | Number of cores to use |

## sg_smooth_cpp *Savitzky-Golay Filter for Matrix (Row-wise)*

### Description

Smooths each row of a matrix using a Savitzky-Golay filter.

### Usage

```
sg_smooth_cpp(x, window_size, order, deriv, n_cores = 1L)
```

### Arguments

| | |
|---|---|
| x | Matrix to smooth row-wise |
| window_size | Size of the filter window (must be odd) |
| order | Polynomial order |
| deriv | Derivative order (0=smooth, 1=first deriv, etc.) |
| n_cores | Number of cores to use |

## summary.dice *Summary function for* dice *objects.*

### Description

Alias of print method.

### Usage

```
## S3 method for class 'dice'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | Object of class dice. |
| ... | Ignored for now. |

summary.ice *Summary function for* ice *objects.*

## Description

Alias of print method.

## Usage

```
## S3 method for class 'ice'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object | Object of class ice. |
| ... | Ignored for now. |

transform_ice_curves_cpp
*Probability Transformation*

## Description

Efficiently applies logodds or probit transformation to a matrix.

## Usage

```
transform_ice_curves_cpp(x, method, n_cores = 1L)
```

## Arguments

| | |
|---|---|
| x | Numeric Matrix (probabilities) |
| method | 1 for centered logodds, 2 for probit |
| n_cores | Number of cores to use |

`WhiteWine` *Data concerning white wine.*

## Description

The WhiteWine data frame has 4898 rows and 12 columns and concerns white wines from a region in Portugal. The response variable, quality, is a wine quality metric, taken to be the median preference score of three blind tasters on a scale of 1-10. The 11 covariates are physicochemical metrics of wine quality such as citric acid content, sulphates, etc.

## Usage

```
data(WhiteWine)
```

## Format

A data frame of 4898 cases on 12 variables.

## Source

K Bache and M Lichman. UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml

# Index

∗ **datasets**
WhiteWine, 21

additivityLineup, 2

backfitter, 3

clusterICE, 4
colSds_cpp, 6

derivative_cpp, 6
dice, *5*, 7, *8*, *14*

ice, *5*, *8*, 9, *16*

melt_ice_curves_cpp, 12

plot.dice, 12
plot.ice, *11*, 14
print.dice, 17
print.ice, *11*, 18

rowCenter_cpp, 18

sg_smooth_cpp, 19
summary.dice, 19
summary.ice, *11*, 20

transform_ice_curves_cpp, 20

WhiteWine, 21