Package ‘isodistrreg’

March 22, 2021

Type Package

Title Isotonic Distributional Regression (IDR)

Version 0.1.0

Date 2021-03-16

Maintainer Alexander Henzi <henzi.alexander@gmail.com>

Description Distributional regression under stochastic order restrictions for numeric and binary response variables and partially ordered covariates. See Henzi, Ziegel, Gneiting (2020) <arXiv:1909.03725>.

License GPL (>= 2)

Depends R (>= 3.6)

Imports Rcpp (>= 1.0.1), osqp, Matrix, utils

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.1.0

Suggests knitr, rmarkdown

URL https://github.com/AlexanderHenzi/isodistrreg

BugReports https://github.com/AlexanderHenzi/isodistrreg/issues

NeedsCompilation yes

Author Alexander Henzi [aut, cre]

Repository CRAN

Date/Publication 2021-03-22 09:20:03 UTC

R topics documented:

`isodistrreg-package` *Isotonic distributional regression (IDR)*

Description

Isotonic distributional Regression (IDR) is a nonparametric method to estimate conditional distributions under monotonicity constraints.

How does it work?

Read the arXiv preprint 'Isotonic Distributional Regression' on https://arxiv.org/abs/1909.03725 or by calling `browseVignettes(package = "isodistrreg")`.

The isodistrreg package

To make probabilistic forecasts with IDR,

- call `idr(y = y, X = X, ...)`, where `y` is the response variable (e.g. weather variable observations) and `X` is a `data.frame` of covariates (e.g. ensemble forecasts).
- use `predict(fit, data)`, where `fit` is the model fit computed with `idr` and `data` is the data based on which you want to make predictions.
- Try `idrbag` for IDR with (su)bagging.

The following pre-defined functions are available to evaluate IDR predictions:

- `cdf` and `qpred` to compute the cumulative distribution function (CDF) and quantile function of IDR predictions.
- `bscore` and `qscore` to calculate Brier scores for probability forecasts for threshold exceedance (e.g. probability of precipitation) and quantile scores (e.g. mean absolute error of median forecast.)
- `crps` to compute the continuous ranked probability score (CRPS).
- `pit` to compute the probability integral transform (PIT).
- `plot` to plot IDR predictive CDFs.

Use the dataset `rain` to test IDR.

References

Alexander Henzi, Johanna F. Ziegel, and Tilmann Gneiting. Isotonic Distributional Regression. arXiv e-prints, art. arXiv:1909.03725, Sep 2019. URL https://arxiv.org/abs/1909.03725.

Examples

```
## A usage example:

# Prepare dataset: Half of the data as training dataset, other half for validation.
# Consult the R documentation (?rain) for details about the dataset.
data(rain)
trainingData <- subset(rain, date <= "2012-01-09")
validationData <- subset(rain, date > "2012-01-09")

# Variable selection: use HRES and the perturbed forecasts P1, ..., P50
varNames <- c("HRES", paste0("P", 1:50))

# Partial orders on variable groups: Usual order of numbers on HRES (group '1') and
# increasing convex order on the remaining variables (group '2').
groups <- setNames(c(1, rep(2, 50)), varNames)
orders <- c("comp" = 1, "icx" = 2)

# Fit IDR to training dataset.
fit <- idr(
  y = trainingData[["obs"]],
  X = trainingData[, varNames],
  groups = groups,
  orders = orders
)

# Make prediction for the first day in the validation data:
firstPrediction <- predict(fit, data = validationData[1, varNames])
plot(firstPrediction)

# Use cdf() and qpred() to make probability and quantile forecasts:

## What is the probability of precipitation?
1 - cdf(firstPrediction, thresholds = 0)

## What are the predicted 10%, 50% and 90% quantiles for precipitation?
qpred(firstPrediction, quantiles = c(0.1, 0.5, 0.9))

# Make predictions for the complete verification dataset and compare IDR calibrated
# forecasts to the raw ensemble (ENS):
predictions <- predict(fit, data = validationData[, varNames])
y <- validationData[["obs"]]

## Continuous ranked probability score (CRPS):
CRPS <- cbind(
  "ens" = crps(validationData[, varNames], y),
```

```
  "IDR" = crps(predictions, y)
)
apply(CRPS, 2, mean)

## Brier score for probability of precipitation:
BS <- cbind(
  "ens" = bscore(validationData[, varNames], thresholds = 0, y),
  "IDR" = bscore(predictions, thresholds = 0, y)
)
apply(BS, 2, mean)

## Quantile score of forecast for 90% quantile:
QS90 <- cbind(
  "ens" = qscore(validationData[, varNames], quantiles = 0.9, y),
  "IDR" = qscore(predictions, quantiles = 0.9, y)
)
apply(QS90, 2, mean)

## Check calibration using (randomized) PIT histograms:
pitEns <- pit(validationData[, varNames], y)
pitIdr <- pit(predictions, y)

hist(pitEns, main = "PIT of raw ensemble forecasts", freq = FALSE)
hist(pitIdr, main = "PIT of IDR calibrated forecasts", freq = FALSE)
```

---

bscore *Brier score for forecast probability of threshold exceedance*

Description

Computes the Brier score of forecast probabilities for exceeding given thresholds.

Usage

```
bscore(predictions, thresholds, y)
```

Arguments

predictions	either an object of class idr (output of predict.idrfit), or a data.frame of numeric variables. In the latter case, the CDF is computed using the empirical distribution of the variables in predictions.
thresholds	numeric vector of thresholds at which the CDF will be evaluated.
y	a numeric vector of obervations of the same length as the number of predictions, or of length 1. In the latter case, y will be used for all predictions.

Details

The Brier score for the event of exceeding a given threshold z is defined as

$$(1\{y > z\} - P(y > z))^2$$

where *y* is the observation and *P(y > z)* the forecast probability for exceeding the threshold z.

Value

A matrix of the Brier scores for the desired thresholds, one column per threshold.

References

Gneiting, T. and Raftery, A. E. (2007), 'Strictly proper scoring rules, prediction, and estimation', Journal of the American Statistical Association 102(477), 359-378

See Also

predict.idrfit, cdf

Examples

```
data("rain")

## Postprocess HRES forecast using data of 3 years

X <- rain[1:(3 * 365), "HRES", drop = FALSE]
y <- rain[1:(3 * 365), "obs"]

fit <- idr(y = y, X = X)

## Compute Brier score for postprocessed probability of precipitation
## forecast using data of the next 2 years (out-of-sample predictions)

data <- rain[(3 * 365 + 1):(5 * 365), "HRES", drop = FALSE]
obs <- rain[(3 * 365 + 1):(5 * 365), "obs"]
predictions <- predict(fit, data = data)
score <- bscore(predictions, thresholds = 0, y = obs)

mean(score)
```

---

cdf — *Cumulative distribution function (CDF) of IDR or raw forecasts*

Description

Evaluate the the cumulative distribution function (CDF) of IDR predictions or of unprocessed forecasts in a `data.frame`.

Usage

```
cdf(predictions, thresholds)

## S3 method for class 'idr'
cdf(predictions, thresholds)

## S3 method for class 'data.frame'
cdf(predictions, thresholds)
```

Arguments

predictions	either an object of class idr (output of predict.idrfit), or a data.frame of numeric variables. In the latter case, the CDF is computed using the empirical distribution of the variables in predictions.
thresholds	numeric vector of thresholds at which the CDF will be evaluated.

Details

The CDFs are considered as piecewise constant stepfunctions: If x are the points where the IDR fitted CDF (or the empirical distribution of the forecasts) has jumps and p the corresponding CDF values, then for x[i] <= x < x[i + 1], the CDF at x is p[i].

Value

A matrix of probabilities giving the evaluated CDFs at the given thresholds, one column for each threshold.

See Also

predict.idrfit qpred, bscore

Examples

```
data("rain")

## Postprocess HRES forecast using data of 3 years

X <- rain[1:(3 * 365), "HRES", drop = FALSE]
y <- rain[1:(3 * 365), "obs"]

fit <- idr(y = y, X = X)

## Compute probability of precipitation given that the HRES forecast is
## 0 mm, 0.5 mm or 1 mm

predictions <- predict(fit, data = data.frame(HRES = c(0, 0.5, 1)))
1 - cdf(predictions, thresholds = 0)
```

`crps` *Continuous ranked probability score (CRPS)*

Description

Computes the CRPS of IDR or raw forecasts.

Usage

```
crps(predictions, y)

## S3 method for class 'idr'
crps(predictions, y)

## S3 method for class 'data.frame'
crps(predictions, y)
```

Arguments

`predictions`	either an object of class `idr` (output of `predict.idrfit`), or a `data.frame` of numeric variables. In the latter case, the CRPS is computed using the empirical distribution of the variables in `predictions`.
`y`	a numeric vector of obervations of the same length as the number of predictions, or of length 1. In the latter case, `y` will be used for all predictions.

Details

This function uses adapted code taken from the function `crps_edf` of the **scoringRules** package.

Value

A vector of CRPS values.

References

Jordan A., Krueger F., Lerch S. (2018). "Evaluating Probabilistic Forecasts with scoringRules." Journal of Statistical Software. Forthcoming.

Gneiting, T. and Raftery, A. E. (2007), 'Strictly proper scoring rules, prediction, and estimation', Journal of the American Statistical Association 102(477), 359-378

See Also

`predict.idrfit`

Examples

```
data("rain")

## Postprocess HRES forecast using data of 3 years

X <- rain[1:(3 * 365), "HRES", drop = FALSE]
y <- rain[1:(3 * 365), "obs"]

fit <- idr(y = y, X = X)

## Compute CRPS of postprocessed HRES forecast using data of the next 2 years
## (out-of-sample predictions)

data <- rain[(3 * 365 + 1):(5 * 365), "HRES", drop = FALSE]
obs <- rain[(3 * 365 + 1):(5 * 365), "obs"]
predictions <- predict(fit, data = data)
idrCrps <- crps(predictions, y = obs)

## Compare this to CRPS of the raw ensemble of all forecasts (high resolution,
## control and 50 perturbed ensemble forecasts)

rawData <- rain[(3 * 365 + 1):(5 * 365), c("HRES", "CTR", paste0("P", 1:50))]
rawCrps <- crps(rawData, y = obs)

c("idr_HRES" = mean(idrCrps), "raw_all" = mean(rawCrps))
```

---

dindexm *Distributional index model (DIM)*

Description

Fits distributional index model with user-specified index function to training dataset. See the examples at the bottom to learn how to specify a distributional single index model.

Usage

```
dindexm(
  formula,
  indexfit,
  data,
  response,
 pars = osqpSettings(verbose = FALSE, eps_abs = 1e-05, eps_rel = 1e-05, max_iter =
    10000L),
  progress = TRUE,
  ...
)
```

Arguments

`formula`	object of class `formula` that describes the index model
`indexfit`	function that fits the index model to training data. Should accept arguments `formula` and `data` and admit a `predict` method. Further arguments in `...` are passed to indexfit. See examples.
`data`	`data.frame` containing the covariates of the index model and the response variable.
`response`	name of the response variable in `data`.
`pars`	parameters for quadratic programming optimization (only relevant for multivariate index functions), set using `osqpSettings`.
`progress`	display progressbar for fitting idr?
`...`	further arguments passed to `indexfit`.

Details

This function fits a distributional index model (DIM) to training data. The DIM assumes that the response is more likely to attain higher values when the values of the index function increases. The index function can be estimated by parametric methods like `lm` or `glm` or also nonparametrically.

The formal mathematical assumption of the DIM is that the conditional CDFs $F_{y|g(X)=g(x)}(z)$ at each fixed threshold z decreases, as *g(x)* increases. Here `y` denotes the response, `x`, `X` are the covariates in `data` and `g` is the index function estimated by `indexfit`.

Estimation is performed in two steps: `indexfit` is applied to `data` to estimate the function `g`. With this estimate, `idr` is applied with the pseudo-covariates `g(x)` and response `y`.

Value

Object of class `dindexm`: A list containing the index model (first component) and the IDR fit on the pseudo-data with the index as covariate (second component).

References

Henzi, A., Kleger, G. R., & Ziegel, J. F. (2020). Distributional (Single) Index Models. arXiv preprint arXiv:2006.09219.

See Also

`idr` for more information on IDR, `predict.dindexfit` for (out-of-sample) predictions based on a model with with `dindexm`.

Examples

```
n <- 1000
X <- data.frame(x1 = rnorm(n), x2 = rnorm(n), x3 = rnorm(n))
y <- rnorm(n, 1 - X[, 1] + X[, 2]^2 / 3 - (1 - X[, 3]) * (1 + X[, 3]) / 2)
data <- cbind(y = y, as.data.frame(X))

## data for out-of-sample prediction
```

```
newX <- data.frame(x1 = rnorm(10), x2 = rnorm(10), x3 = rnorm(10))

## linear regression model for index
model <- dindexm(
  formula = y ~ poly(x1, degree = 2) + poly(x2, degree = 2) +
    poly(x3, degree = 2),
  indexfit = lm,
  response = "y",
  data = data
)
pred <- predict(model, data = newX)

## plot
plot(pred, 1, main = "LM based DIM")
grd <- pred[[1]]$points
trueCdf <- pnorm(
  grd,
  1 - newX[1, 1] + newX[1, 2]^2 / 3 - (1 - newX[1, 3]) * (1 + newX[1, 3]) / 2
)
points(grd, trueCdf, type = "l", col = 2)
```

`idr` *Fit IDR to training data*

Description

Fits isotonic distributional regression (IDR) to a training dataset.

Usage

```
idr(y, X, groups = setNames(rep(1, ncol(X)), colnames(X)), orders =
  c("comp" = 1), stoch = "sd", pars = osqpSettings(verbose = FALSE, eps_abs =
  1e-5, eps_rel = 1e-5, max_iter = 10000L), progress = TRUE)
```

Arguments

`y`	numeric vector (the response variable).
`X`	data frame of numeric or ordered factor variables (the regression covariates).
`groups`	named vector of length `ncol(X)` denoting groups of variables that are to be ordered with the same order (see 'Details'). Only relevant if `X` contains more than one variable. The same names as in `X` should be used.
`orders`	named vector giving for each group in `groups` the order that will be applied to this group. Only relevant if `X` contains more than one variable. The names of `orders` give the order, the entries give the group labels. Available options: `"comp"` for componentwise order, `"sd"` for stochastic dominance, `"icx"` for increasing convex order (see 'Details). Default is `"comp"` for all variables. The `"sd"` and `"icx"` orders can only be used with numeric variables, but not with ordered factors.

`stoch`	stochastic order constraint used for estimation. Default is `"sd"` for first order stochastic dominance. Use `"hazard"` for hazard rate order (experimental).
`pars`	parameters for quadratic programming optimization (only relevant if `X` has more than one column), set using `osqpSettings`.
`progress`	display progressbar (`TRUE`, `FALSE` or `1`, `0`)?

Details

This function computes the isotonic distributional regression (IDR) of a response *y* on on one or more covariates *X*. IDR estimates the cumulative distribution function (CDF) of *y* conditional on *X* by monotone regression, assuming that *y* is more likely to take higher values, as *X* increases. Formally, IDR assumes that the conditional CDF $F_{y|X=x}(z)$ at each fixed threshold z decreases, as *x* increases, or equivalently, that the exceedance probabilities for any threshold `z` $P(y > z|X = x)$ increase with *x*.

The conditional CDFs are estimated at each threshold in `unique(y)`. This is the set where the CDFs may have jumps. If `X` contains more than one variable, the CDFs are estimated by calling `solve_osqp` from the package **osqp** `length(unique(y))` times. This might take a while if the training dataset is large.

Use the argument `groups` to group *exchangeable* covariates. Exchangeable covariates are indistinguishable except from the order in which they are labelled (e.g. ensemble weather forecasts, repeated measurements under the same measurement conditions).

The following orders are available to perform the monotone regression in IDR:

- Componentwise order (`"comp"`): A covariate vector `x1` is greater than `x2` if `x1[i] >= x2[i]` holds for all components `i`. This is the *standard order used in multivariate monotone regression* and *should not be used for exchangeable variables (e.g. perturbed ensemble forecasts).*
- Stochastic dominance (`"sd"`): `x1` is greater than `x2` in the stochastic order, if the (empirical) distribution of the elements of `x1` is greater than the distribution of the elements of `x2` (in first order) stochastic dominance. The `"sd"` order is invariant under permutations of the grouped variables and therefore *suitable for exchangeable covariables.*
- Increasing convex order (`"icx"`): The `"icx"` order can be used for groups of exchangeable variables. It should be used if the variables have increasing variability, when their mean increases (e.g. precipitation forecasts or other variables with right-skewed distributions). More precisely, `"icx"` uses the increasing convex stochastic order on the empirical distributions of the grouped variables.

Value

An object of class `"idrfit"` containing the following components:

`X`	data frame of all distinct covariate combinations used for the fit.
`y`	list of all observed responses in the training data for given covariate combinations in `X`.
`cdf`	matrix containing the estimated CDFs, one CDF per row, evaluated at `thresholds` (see next point). The CDF in the `i`th row corredponds to the estimated conditional distribution of the response given the covariates values in `X[i,]`.

thresholds	the thresholds at which the CDFs in cdf are evaluated. The entries in cdf[,j] are the conditional CDFs evaluated at thresholds[j].
groups, orders	the groups and orders used for estimation.
diagnostic	list giving a bound on the precision of the CDF estimation (the maximal downwards-step in the CDF that has been detected) and the fraction of CDF estimations that were stopped at the iteration limit max_iter. Decrease the parameters eps_abs and/or eps_rel or increase max_iter in pars to improve the precision. See osqpSettings for more optimization parameters.
indices	the row indices of the covariates in X in the original training dataset (used for in-sample predictions with predict.idrfit).
constraints	(in multivariate IDR, NULL otherwise) matrices giving the order constraints for optimization. Used in predict.idrfit.

Note

The function idr is only intended for fitting IDR model for a training dataset and storing the results for further processing, but not for prediction or evaluation, which is done using the output of predict.idrfit.

Author(s)

Code for the Pool-Adjacent Violators Algorithm (PAVA) is adapted from R code by Lutz Duembgen (available on https://www.imsv.unibe.ch/about_us/files/lutz_duembgen/software/index_eng.html).

References

Henzi, A., Moesching, A., & Duembgen, L. (2020). Accelerating the pool-adjacent-violators algorithm for isotonic distributional regression. arXiv preprint arXiv:2006.05527.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. Mathematical Programming Computation, 1-36.

Bartolomeo Stellato, Goran Banjac, Paul Goulart and Stephen Boyd (2019). osqp: Quadratic Programming Solver using the 'OSQP' Library. R package version 0.6.0.3. https://CRAN.R-project.org/package=osqp

See Also

The S3 method predict.idrfit for predictions based on an IDR fit.

Examples

```
data("rain")

## Fit IDR to data of 185 days using componentwise order on HRES and CTR and
## increasing convex order on perturbed ensemble forecasts (P1, P2, ..., P50)

varNames <- c("HRES", "CTR", paste0("P", 1:50))
X <- rain[1:185, varNames]
```

```
y <- rain[1:185, "obs"]

## HRES and CTR are group '1', with componentwise order "comp", perturbed
## forecasts P1, ..., P50 are group '2', with "icx" order

groups <- setNames(c(1, 1, rep(2, 50)), varNames)
orders <- c("comp" = 1, "icx" = 2)

fit <- idr(y = y, X = X, orders = orders, groups = groups)
fit
```

---

idrbag *Compute IDR predictions with (su)bagging*

Description

Computes IDR predictions with bootstrap aggregating (bagging) or subsample aggregation (subagging).

Usage

```
idrbag(y, X, groups = setNames(rep(1, ncol(X)), colnames(X)), orders =
  c("comp" = 1), stoch = "sd", pars = osqpSettings(verbose = FALSE, eps_abs =
  1e-5, eps_rel = 1e-5, max_iter = 10000L), progress = TRUE, newdata,
  digits = 3, interpolation = "linear", b, p, replace = FALSE, grid = NULL)
```

Arguments

`y`	numeric vector (the response variable).
`X`	data frame of numeric or ordered factor variables (the regression covariates).
`groups`	named vector of length `ncol(X)` denoting groups of variables that are to be ordered with the same order (see 'Details'). Only relevant if `X` contains more than one variable. The same names as in `X` should be used.
`orders`	named vector giving for each group in `groups` the order that will be applied to this group. Only relevant if `X` contains more than one variable. The names of `orders` give the order, the entries give the group labels. Available options: `"comp"` for componentwise order, `"sd"` for stochastic dominance, `"icx"` for increasing convex order (see 'Details). Default is `"comp"` for all variables. The `"sd"` and `"icx"` orders can only be used with numeric variables, but not with ordered factors.
`stoch`	stochastic order constraint used for estimation. Default is `"sd"` for first order stochastic dominance. Use `"hazard"` for hazard rate order (experimental).
`pars`	parameters for quadratic programming optimization (only relevant if `X` has more than one column), set using `osqpSettings`.
`progress`	display progressbar (`TRUE`, `FALSE` or `1`, `0`)?

newdata	data.frame containing variables with which to predict. Ordered factor variables are converted to numeric for computation, so ensure that the factor levels are identical in newdata and in X.
digits	number of decimal places for the predictive CDF.
interpolation	interpolation method for univariate data. Default is "linear". Any other argument will select midpoint interpolation (see 'Details' in predict.idrfit). Has no effect for multivariate IDR.
b	number of (su)bagging samples.
p	size of (su)bagging samples relative to training data.
replace	draw samples with (TRUE, 1) or without (FALSE, 0) replacement?
grid	grid on which the predictive CDFs are evaluated. Default are the unique values of y.

Details

This function draws b times a random subsample of size ceiling(nrow(X)*p)) from the training data, fits IDR to each subsample, computes predictions for the new data supplied in newdata, and averages the predictions derived from the b subsamples. There are no default values for b and p.

Value

A list of predictions, see predict.idrfit.

---

pit *Probability integral transform (PIT)*

Description

Computes the probability integral transform (PIT) of IDR or raw forecasts.

Usage

```
pit(predictions, y, randomize = TRUE, seed = NULL)

## S3 method for class 'idr'
pit(predictions, y, randomize = TRUE, seed = NULL)

## S3 method for class 'data.frame'
pit(predictions, y, randomize = TRUE, seed = NULL)
```

Arguments

predictions	either an object of class idr (output of predict.idrfit), or a data.frame of numeric variables. In the latter case, the PIT is computed using the empirical distribution of the variables in predictions.
y	a numeric vector of obervations of the same length as the number of predictions.
randomize	PIT values should be randomized at discontinuity points of the predictive CDF (e.g. at zero for precipitation forecasts). Set randomize = TRUE to randomize.
seed	argument to set.seed for random number generation (if randomize is TRUE).

Value

Vector of PIT values.

References

Gneiting, T., Balabdaoui, F. and Raftery, A. E. (2007), 'Probabilistic forecasts, calibration and sharpness', Journal of the Royal Statistical Society: Series B (Statistical Methodology) 69(2), 243-268.

See Also

predict.idrfit

Examples

```
data("rain")
require("graphics")

## Postprocess HRES forecast using data of 4 years

X <- rain[1:(4 * 365), "HRES", drop = FALSE]
y <- rain[1:(4 * 365), "obs"]

fit <- idr(y = y, X = X)

## Assess calibration of the postprocessed HRES forecast using data of next 4
## years and compare to calibration of the raw ensemble

data <- rain[(4 * 365 + 1):(8 * 365), "HRES", drop = FALSE]
obs <- rain[(4 * 365 + 1):(8 * 365), "obs"]
predictions <- predict(fit, data = data)
idrPit <- pit(predictions, obs, seed = 123)

rawData <- rain[(4 * 365 + 1):(8 * 365), c("HRES", "CTR", paste0("P", 1:50))]
rawPit <- pit(rawData, obs, seed = 123)

hist(idrPit, xlab = "Probability Integral Transform",
  ylab = "Density", freq = FALSE, main = "Postprocessed HRES")
hist(rawPit, xlab = "Probability Integral Transform",
  ylab = "Density", freq = FALSE, main = "Raw ensemble")
```

plot.idr *Plot IDR predictions*

Description

Plot an IDR predictive CDF.

Usage

```
## S3 method for class 'idr'
plot(
  x,
  index = 1,
  bounds = TRUE,
  col.cdf = "black",
  col.bounds = "blue",
  lty.cdf = 1,
  lty.bounds = 3,
  xlab = "Threshold",
  ylab = "CDF",
  main = "IDR predictive CDF",
  ...
)
```

Arguments

`x`	object of class `idr` (output of `predict.idrfit`).
`index`	index of the prediction in `x` for which a plot is desired.
`bounds`	whether the bounds should be plotted or not (see `predict.idrfit` for details about the meaning of the bounds).
`col.cdf`	color of the predictive CDF.
`col.bounds`	color of the bounds.
`lty.cdf`	linetype of the predictive CDF.
`lty.bounds`	linetype of the CDF bounds.
`xlab`	label for x axis.
`ylab`	label for y axis.
`main`	main title.
`...`	further arguments to `plot.stepfun` or `plot`.

Value

The data based on which the plot is drawn (returned invisible).

See Also

predict.idrfit

Examples

```
data("rain")
require("graphics")

## Postprocess HRES and CTR forecast using data of 2 years

X <- rain[1:(2 * 365), c("HRES", "CTR"), drop = FALSE]
y <- rain[1:(2 * 365), "obs"]

## Fit IDR and plot the predictive CDF when the HRES forecast is 1 mm and
## CTR is 0 mm

fit <- idr(y = y, X = X)
pred <- predict(fit, data = data.frame(HRES = 1, CTR = 0))
plot(pred)
```

---

predict.dindexfit *Predict method for distributional index model (DIM)*

Description

Prediction based on distributional index model fit.

Usage

```
## S3 method for class 'dindexfit'
predict(
  object,
  data = NULL,
  digits = 3,
  interpolation = "linear",
  asplitAvail = TRUE,
  ...
)
```

Arguments

object	DIM fit (object of class "dindexfit").
data	optional data.frame containing variables with which to predict. In-sample predictions are returned if this is omitted.
digits	number of decimal places for the predictive CDF.
interpolation	interpolation method for univariate index Default is "linear". Any other argument will select midpoint interpolation (see 'Details' in predict.idrfit). Has no effect for multivariate index function.

`asplitAvail`	use `asplit` for splitting arrays (default is `TRUE`). Set to `FALSE` for R Versions < 3.6, where `asplit` is not available.
`...`	further arguments passed to the index prediction function.

Value

A list of predictions, as for `predict.idrfit`.

See Also

Examples in `dindexm`.

`predict.idrfit` *Predict method for IDR fits*

Description

Prediction based on IDR model fit.

Usage

```
## S3 method for class 'idrfit'
predict(object, data = NULL, digits = 3, interpolation = "linear", ...)
```

Arguments

`object`	IDR fit (object of class `"idrfit"`).
`data`	optional `data.frame` containing variables with which to predict. In-sample predictions are returned if this is omitted. Ordered factor variables are converted to numeric for computation, so ensure that the factor levels are identical in `data` and the training data for `fit`.
`digits`	number of decimal places for the predictive CDF.
`interpolation`	interpolation method for univariate data. Default is `"linear"`. Any other argument will select midpoint interpolation (see 'Details'). Has no effect for multivariate IDR.
`...`	included for generic function consistency.

Details

If the variables `x = data[j,]` for which predictions are desired are already contained in the training dataset `X` for the fit, `predict.idrfit` returns the corresponding in-sample prediction. Otherwise monotonicity is used to derive upper and lower bounds for the predictive CDF, and the predictive CDF is a pointwise average of these bounds. For univariate IDR with a numeric covariate, the predictive CDF is computed by linear interpolation. Otherwise, or if `interpolation != "linear"`, midpoint interpolation is used, i.e. default weights of `0.5` for both the lower and the upper bound.

If the lower and the upper bound on the predictive cdf are far apart (or trivial, i.e. constant 0 or constant 1), this indicates that the prediction based on x is uncertain because either the training dataset is too small or only few similar variable combinations as in x have been observed in the training data. However, *the bounds on the predictive CDF are not prediction intervals and should not be interpreted as such. They only indicate the uncertainty of out-of-sample predictions for which the variables are not contained in the training data.*

If the new variables x are greater than all X[i,] in the selected order(s), the lower bound on the cdf is trivial (constant 0) and the upper bound is taken as predictive cdf. The upper bound on the cdf is trivial (constant 1) if x is smaller than all X[i,]. If x is not comparable to any row of X in the given order, a prediction based on the training data is not possible. In that case, the default forecast is the empirical distribution of y in the training data.

Value

A list of predictions. Each prediction is a data.frame containing the following variables:

points	the points where the predictive CDF has jumps.
cdf	the estimated CDF evaluated at the points.
lower, upper	(only for out-of-sample predictions) bounds for the estimated CDF, see 'Details' above.

The output has the attribute incomparables, which gives the indices of all predictions for which the climatological forecast is returned because the forecast variables are not comparable to the training data.

See Also

idr to fit IDR to training data.

cdf, qpred to evaluate the CDF or quantile function of IDR predictions.

bscore, qscore, crps, pit to compute Brier scores, quantile scores, the CRPS and the PIT of IDR predictions.

plot to plot IDR predictive CDFs.

Examples

```
data("rain")

## Fit IDR to data of 185 days using componentwise order on HRES and CTR and
## increasing convex order on perturbed ensemble forecasts (P1, P2, ..., P50)

varNames <- c("HRES", "CTR", paste0("P", 1:50))
X <- rain[1:185, varNames]
y <- rain[1:185, "obs"]

## HRES and CTR are group '1', with componentwise order "comp", perturbed
## forecasts P1, ..., P50 are group '2', with "icx" order

groups <- setNames(c(1, 1, rep(2, 50)), varNames)
orders <- c("comp" = 1, "icx" = 2)
```

```
fit <- idr(y = y, X = X, orders = orders, groups = groups)

## Predict for day 186
predict(fit, data = rain[186, varNames])
```

---

qpred *Quantile function of IDR or raw forecasts*

Description

Evaluate the the quantile function of IDR predictions or of unprocessed forecasts in a `data.frame`.

Usage

```
qpred(predictions, quantiles)

## S3 method for class 'idr'
qpred(predictions, quantiles)

## S3 method for class 'data.frame'
qpred(predictions, quantiles)
```

Arguments

`predictions`	either an object of class `idr` (output of `predict.idrfit`), or a `data.frame` of numeric variables. In the latter case, quantiles are computed using the empirical distribution of the variables in `predictions`.
`quantiles`	numeric vector of desired quantiles.

Details

The quantiles are defined as lower quantiles, that is,

$$q(u) = inf(x : cdf(x) >= u).$$

Value

A matrix of forecasts for the desired quantiles, one column per quantile.

See Also

`predict.idrfit`, `cdf`, `qscore`

Examples

```
data("rain")

## Postprocess HRES forecast using data of 3 years

X <- rain[1:(3 * 365), "HRES", drop = FALSE]
y <- rain[1:(3 * 365), "obs"]

fit <- idr(y = y, X = X)

## Compute 95%-quantile forecast given that the HRES forecast is
## 2.5 mm, 5 mm or 10 mm

predictions <- predict(fit, data = data.frame(HRES = c(2.5, 5, 10)))
qpred(predictions, quantiles = 0.95)
```

---

qscore — *Quantile scores for IDR or raw forecasts*

Description

Computes quantile scores of IDR quantile predictions or of quantile predictions from raw forecasts in a `data.frame`.

Usage

```
qscore(predictions, quantiles, y)
```

Arguments

`predictions`	either an object of class `idr` (output of `predict.idrfit`), or a `data.frame` of numeric variables. In the latter case, quantiles are computed using the empirical distribution of the variables in `predictions`.
`quantiles`	numeric vector of desired quantiles.
`y`	a numeric vector of obervations of the same length as the number of predictions, or of length 1. In the latter case, `y` will be used for all predictions.

Details

The quantile score of a forecast *x* for the *u*-quantile is defined as

$$2(1x > y - u)(x - y),$$

where *y* is the observation. For *u* = *1/2*, this equals the mean absolute error of the median forecast.

Value

A matrix of the quantile scores for the desired quantiles, one column per quantile.

References

Gneiting, T. and Raftery, A. E. (2007), 'Strictly proper scoring rules, prediction, and estimation', Journal of the American Statistical Association 102(477), 359-378

See Also

predict.idrfit, qpred

Examples

```
data("rain")

## Postprocess HRES forecast using data of 3 years

X <- rain[1:(3 * 365), "HRES", drop = FALSE]
y <- rain[1:(3 * 365), "obs"]

fit <- idr(y = y, X = X)

## Compute mean absolute error of the median postprocessed forecast using
## data of the next 2 years (out-of-sample predictions) and compare to raw
## HRES forecast

data <- rain[(3 * 365 + 1):(5 * 365), "HRES", drop = FALSE]
obs <- rain[(3 * 365 + 1):(5 * 365), "obs"]

predictions <- predict(fit, data = data)
idrMAE <- mean(qscore(predictions, 0.5, obs))
rawMAE <- mean(qscore(data, 0.5, obs))

c("idr" = idrMAE, "raw" = rawMAE)
```

---

rain *Frankfurt airport precipitation data*

Description

Accumulated 06-30 hour precipitation observations and operational ECMWF ensemble forecasts for Frankfurt airport, Germany. The observations are airport station observations (WMO station index 10637), the forecasts are gridded forecasts for the 0.25 degrees latitude/longitude box containing the station. Dates range from 2007-01-01 to 2017-01-01, days with missing values have been removed.

Usage

```
data("rain")
```

Format

A data frame with 3617 rows. The first column gives the dates, the second column are the observations. The remaining columns are the ensemble forecasts (high resolution HRES, 50 perturbed forecasts P1 to P50 and the control forecast CTR for the perturbed forecasts). The units of the forecasts and observations are mm/m^2.

Source

Observations: http://www.ogimet.com/synops.phtml.en

Forecasts: available on TIGGE https://confluence.ecmwf.int/display/TIGGE/TIGGE+archive

References

Bougeault et al. (2010) The THORPEX Interactive Grand Global Ensemble. Bull. Amer. Meteor. Soc., 91, 1059-1072.

Swinbank et al. (2016) The TIGGE project and its achievements. Bull. Amer. Meteor. Soc., 97, 49-67.

Index