Package: NNS (via r-universe)

June 30, 2026

Type Package

Title Nonlinear Nonparametric Statistics

Version 13.0

Date 2026-06-30

Maintainer Fred Viole <ovvo.open.source@gmail.com>

Description NNS (Nonlinear Nonparametric Statistics) leverages partial moments – the fundamental elements of variance that asymptotically approximate the area under f(x) – to provide a robust foundation for nonlinear analysis while maintaining linear equivalences. Designed for real-world data that violates symmetry, linearity, or distributional assumptions, NNS delivers a comprehensive suite of advanced statistical techniques, including: Numerical integration, Numerical differentiation, Clustering, Correlation, Dependence, Causal analysis, ANOVA, Regression, Classification, Seasonality, Autoregressive modeling, Normalization, Stochastic superiority / dominance and Advanced Monte Carlo sampling. All routines based on: Viole, F. and Nawrocki, D. (2013), Nonlinear Nonparametric Statistics: Using Partial Moments (ISBN: 1490523995, Second edition: <https://ovvo-financial.github.io/NNS/book/>).

BugReports https://github.com/OVVO-Financial/NNS/issues

License GPL-3

URL https://github.com/OVVO-Financial/NNS

Depends R (>= 3.6.0)

Imports Rcpp, RcppParallel

Suggests knitr, rgl, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

LinkingTo Rcpp, RcppParallel

SystemRequirements GNU make

Config/testthat/edition 3

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation yes

Author Fred Viole [aut, cre], Roberto Spadim [ctb], Rasheed Khoshnaw [ctb]

Config/pak/sysreqs make

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

Date/Publication 2026-06-30 19:12:21 UTC

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

RemoteRef HEAD

RemoteSha c96cd5bd4adb1fa9816aa05d70cbbfc0e7294717

Contents

`Co.LPM` *Co-Lower Partial Moment*

Description

Computes the co-lower partial moment (lower-left quadrant 4) between two equal-length numeric vectors at any degree and target.

Usage

```
Co.LPM(degree_lpm, x, y, target_x, target_y, degree_y = NULL)
```

Arguments

`degree_lpm`	numeric; degree for x ("degree_x"). degree = 0 gives frequency, degree = 1 gives area.
`x`	numeric vector of observations.
`y`	numeric vector of the same length as x.
`target_x`	numeric vector; thresholds for x (defaults to mean(x)).
`target_y`	numeric vector; thresholds for y (defaults to mean(y)).
`degree_y`	numeric; optional degree for y. If omitted, ‘degree_lpm‘ is used for both x and y.

Value

Numeric vector of co-LPM values.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. & Nawrocki, D. (2013) *Nonlinear Nonparametric Statistics: Using Partial Moments* (ISBN:1490523995)

Examples

```
set.seed(123)
x <- rnorm(100); y <- rnorm(100)
Co.LPM(0, x, y, mean(x), mean(y))
```

Co.LPM_nD *Co-Lower Partial Moment nD*

Description

This function generates an n-dimensional co-lower partial moment (n >= 2) for any degree or target.

Usage

```
Co.LPM_nD(data, target, degree = 0, norm = TRUE)
```

Arguments

data	A numeric matrix with observations in rows and variables in columns.
target	A numeric vector, length equal to ncol(data).
degree	numeric; degree for lower deviations (0 = frequency, 1 = area).
norm	logical; if TRUE (default) normalize to the maximum observed value ($\rightarrow$ [0,1]), otherwise return the raw moment.

Value

Numeric; the n-dimensional co-lower partial moment.

Examples

```
## Not run:
mat <- matrix(rnorm(200), ncol = 4)
Co.LPM_nD(mat, rep(0, ncol(mat)), degree = 1, norm = FALSE)

## End(Not run)
```

`Co.UPM` *Co-Upper Partial Moment*

Description

Computes the co-upper partial moment (upper-right quadrant 1) between two equal-length numeric vectors at any degree and target.

Usage

```
Co.UPM(degree_upm, x, y, target_x, target_y, degree_y = NULL)
```

Arguments

`degree_upm`	numeric; degree for x ("degree_x"). degree = 0 gives frequency, degree = 1 gives area.
`x`	numeric vector of observations.
`y`	numeric vector of the same length as x.
`target_x`	numeric vector; thresholds for x (defaults to mean(x)).
`target_y`	numeric vector; thresholds for y (defaults to mean(y)).
`degree_y`	numeric; optional degree for y. If omitted, ‘degree_upm‘ is used for both x and y.

Value

Numeric vector of co-UPM values.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. & Nawrocki, D. (2013) *Nonlinear Nonparametric Statistics: Using Partial Moments* (ISBN:1490523995)

Examples

```
set.seed(123)
x <- rnorm(100); y <- rnorm(100)
Co.UPM(0, x, y, mean(x), mean(y))
```

Co.UPM_nD *Co-Upper Partial Moment nD*

Description

This function generates an n-dimensional co-upper partial moment (n >= 2) for any degree or target.

Usage

```
Co.UPM_nD(data, target, degree = 0, norm = TRUE)
```

Arguments

data	A numeric matrix with observations in rows and variables in columns.
target	A numeric vector, length equal to ncol(data).
degree	numeric; degree for upper deviations (0 = frequency, 1 = area).
norm	logical; if TRUE (default) normalize to the maximum observed value ($\rightarrow$ [0,1]), otherwise return the raw moment.

Value

Numeric; the n-dimensional co-upper partial moment.

Examples

```
## Not run:
mat <- matrix(rnorm(200), ncol = 4)
Co.UPM_nD(mat, rep(0, ncol(mat)), degree = 1, norm = FALSE)

## End(Not run)
```

D.LPM *Divergent-Lower Partial Moment*

Description

Computes the divergent lower partial moment (lower-right quadrant 3) between two equal-length numeric vectors.

Usage

```
D.LPM(degree_lpm, degree_upm, x, y, target_x, target_y)
```

Arguments

degree_lpm	numeric; LPM degree = 0 gives frequency, = 1 gives area.
degree_upm	numeric; UPM degree = 0 gives frequency, = 1 gives area.
x	numeric vector of observations.
y	numeric vector of the same length as x.
target_x	numeric vector; thresholds for x (defaults to mean(x)).
target_y	numeric vector; thresholds for y (defaults to mean(y)).

Value

Numeric vector of divergent LPM values.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. & Nawrocki, D. (2013) *Nonlinear Nonparametric Statistics: Using Partial Moments* (ISBN:1490523995)

Examples

```
set.seed(123)
x <- rnorm(100); y <- rnorm(100)
D.LPM(0, 0, x, y, mean(x), mean(y))
```

D.UPM *Divergent-Upper Partial Moment*

Description

Computes the divergent upper partial moment (upper-left quadrant 2) between two equal-length numeric vectors.

Usage

```
D.UPM(degree_lpm, degree_upm, x, y, target_x, target_y)
```

Arguments

degree_lpm	numeric; LPM degree = 0 gives frequency, = 1 gives area.
degree_upm	numeric; UPM degree = 0 gives frequency, = 1 gives area.
x	numeric vector of observations.
y	numeric vector of the same length as x.
target_x	numeric vector; thresholds for x (defaults to mean(x)).
target_y	numeric vector; thresholds for y (defaults to mean(y)).

Value

Numeric vector of divergent UPM values.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. & Nawrocki, D. (2013) *Nonlinear Nonparametric Statistics: Using Partial Moments* (ISBN:1490523995)

Examples

```
set.seed(123)
x <- rnorm(100); y <- rnorm(100)
D.UPM(0, 0, x, y, mean(x), mean(y))
```

DPM_nD *Divergent Partial Moment nD*

Description

This function generates the aggregate n-dimensional divergent partial moment (n >= 2) for any degree or target.

Usage

```
DPM_nD(data, target, degree = 0, norm = TRUE)
```

Arguments

data	A numeric matrix with observations in rows and variables in columns.
target	A numeric vector, length equal to ncol(data).
degree	numeric; degree for upper deviations (0 = frequency, 1 = area).
norm	logical; if TRUE (default) normalize to the maximum observed value ($\rightarrow$ [0,1]), otherwise return the raw moment.

Value

Numeric; the n-dimensional divergent partial moment.

Examples

```
## Not run:
mat <- matrix(rnorm(200), ncol = 4)
DPM_nD(mat, rep(0, ncol(mat)), degree = 1, norm = FALSE)

## End(Not run)
```

dy.d_ *Partial Derivative dy/d_[wrt]*

Description

Returns the numerical partial derivative of y with respect to [wrt] any regressor for a point of interest. Finite difference method is used with NNS.reg estimates as `f(x + h)` and `f(x - h)` values.

Usage

```
dy.d_(x, y, wrt, eval.points = "obs", mixed = FALSE, messages = TRUE)
```

Arguments

`x`	a numeric matrix or data frame.
`y`	a numeric vector with compatible dimensions to `x`.
`wrt`	integer; Selects the regressor to differentiate with respect to (vectorized).
`eval.points`	numeric or options: ("obs", "apd", "mean", "median", "last"); Regressor points to be evaluated.

- Numeric values must be in matrix or data.frame form to be evaluated for each regressor, otherwise, a vector of points will evaluate only at the `wrt` regressor. See examples for use cases.
- Set to `(eval.points = "obs")` (default) to find the average partial derivative at every observation of the variable with respect to *for specific tuples of given observations.*
- Set to `(eval.points = "apd")` to find the average partial derivative at every observation of the variable with respect to *over the entire distribution of other regressors.*
- Set to `(eval.points = "mean")` to find the partial derivative at the mean of value of every variable.
- Set to `(eval.points = "median")` to find the partial derivative at the median value of every variable.
- Set to `(eval.points = "last")` to find the partial derivative at the last observation of every value (relevant for time-series data).

`mixed`	logical; `FALSE` (default) If mixed derivative is to be evaluated, set `(mixed = TRUE)`.
`messages`	logical; `TRUE` (default) Prints status messages.

Value

Returns column-wise matrix of wrt regressors:

- `dy.d_(...)[, wrt]$First` the 1st derivative
- `dy.d_(...)[, wrt]$Second` the 2nd derivative
- `dy.d_(...)[, wrt]$Mixed` the mixed derivative (for two independent variables only).

Note

For binary regressors, it is suggested to use `eval.points = seq(0, 1, .05)` for a better resolution around the midpoint.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Vinod, H. and Viole, F. (2020) "Comparing Old and New Partial Derivative Estimates from Nonlinear Nonparametric Regressions" doi:10.2139/ssrn.3681104

Examples

```
## Not run:
set.seed(123) ; x_1 <- runif(1000) ; x_2 <- runif(1000) ; y <- x_1 ^ 2 * x_2 ^ 2
B <- cbind(x_1, x_2)

## To find derivatives of y wrt 1st regressor for specific points of both regressors
dy.d_(B, y, wrt = 1, eval.points = t(c(.5, 1)))

## To find average partial derivative of y wrt 1st regressor,
only supply 1 value in [eval.points], or a vector of [eval.points]:
dy.d_(B, y, wrt = 1, eval.points = .5)

dy.d_(B, y, wrt = 1, eval.points = fivenum(B[,1]))


## To find average partial derivative of y wrt 1st regressor,
for every observation of 1st regressor:
apd <- dy.d_(B, y, wrt = 1, eval.points = "apd")
plot(B[,1], apd[,1]$First)

## 95% Confidence Interval to test if 0 is within
### Lower CI
LPM.VaR(.025, 0, apd[,1]$First)

### Upper CI
UPM.VaR(.025, 0, apd[,1]$First)

## End(Not run)
```

dy.dx *Partial Derivative dy/dx*

Description

Returns the numerical partial derivative of y wrt x for a point of interest.

Usage

```
dy.dx(x, y, eval.point = NULL)
```

Arguments

x	a numeric vector.
y	a numeric vector.
eval.point	numeric or ("overall"); x point to be evaluated, must be provided. Defaults to (eval.point = NULL). Set to (eval.point = "overall") to find an overall partial derivative estimate (1st derivative only).

Value

Returns a data.frame of eval.point along with both 1st and 2nd derivative.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Vinod, H. and Viole, F. (2017) "Nonparametric Regression Using Clusters" doi:10.1007/s10614-01797135

Examples

```
## Not run:
x <- seq(0, 2 * pi, pi / 100) ; y <- sin(x)
dy.dx(x, y, eval.point = 1.75)

# First derivative
dy.dx(x, y, eval.point = 1.75)$first.derivative

# Second derivative
dy.dx(x, y, eval.point = 1.75)$second.derivative

# Vector of derivatives
dy.dx(x, y, eval.point = c(1.75, 2.5))
```

```
## End(Not run)
```

LPM *Lower Partial Moment*

Description

This function generates a univariate lower partial moment for any degree or target.

Usage

```
LPM(degree, target, variable, excess_ret = FALSE)
```

Arguments

degree	numeric; (degree = 0) is frequency, (degree = 1) is area.
target	numeric; Set to target = mean(variable) for classical equivalences, but does not have to be. When excess_ret = FALSE, this can be a scalar or a vectorized target for the standard partial moment calculation. When excess_ret = TRUE, it is interpreted element-wise as the benchmark/threshold relative to variable.
variable	a numeric vector. data.frame or list type objects are not permissible.
excess_ret	logical; FALSE (default). If TRUE, switches from the standard vectorized-target partial moment to an element-wise excess-deviation calculation. For LPM, this computes pmax(target - variable, 0) raised to degree and averaged. In this mode, target must have length 1 or the same length as variable.

Value

LPM of variable

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
set.seed(123)
x <- rnorm(100)
LPM(0, mean(x), x)
```

`LPM.ratio` *Lower Partial Moment Ratio*

Description

This function generates a standardized univariate lower partial moment of any non-negative degree for a given target.

Usage

```
LPM.ratio(degree, target, variable)
```

Arguments

`degree`	numeric; degree = 0 gives frequency (CDF), degree = 1 gives area.
`target`	numeric vector; threshold(s). Defaults to mean(variable).
`variable`	numeric vector or data-frame column to evaluate.

Value

Numeric vector of standardized lower partial moments.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. & Nawrocki, D. (2013) *Nonlinear Nonparametric Statistics: Using Partial Moments* (ISBN:1490523995)

Viole, F. (2017) Continuous CDFs and ANOVA with NNS. doi:10.2139/ssrn.3007373

Examples

```
  set.seed(123)
  x <- rnorm(100)
  LPM.ratio(0, mean(x), x)
## Not run:
  plot(sort(x), LPM.ratio(0, sort(x), x))
  plot(sort(x), LPM.ratio(1, sort(x), x))

## End(Not run)
```

LPM.VaR *LPM VaR*

Description

Generates a value at risk (VaR) quantile based on the Lower Partial Moment ratio.

Usage

```
LPM.VaR(percentile, degree, x)
```

Arguments

percentile	numeric [0, 1]; The percentile for left-tail VaR.
degree	integer; (degree = 0) for discrete distributions, (degree = 1) for continuous distributions.
x	a numeric vector.

Value

Returns a numeric value representing the point at which "percentile" of the area of x is below.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100)

## For 5th percentile, left-tail
LPM.VaR(0.05, 0, x)

## End(Not run)
```

NNS.ANOVA *NNS ANOVA: Nonparametric Analysis of Variance*

Description

Performs a distribution-free ANOVA using partial-moment statistics to assess differences between control and treatment groups. Depending on the setting of `means.only`, the procedure tests either differences in central tendency (means or medians) or differences across the full empirical distributions.

Usage

```
NNS.ANOVA(
  control,
  treatment,
  means.only = FALSE,
  medians = FALSE,
  confidence.interval = 0.95,
  tails = "Both",
  pairwise = FALSE,
  plot = TRUE,
  robust = FALSE
)
```

Arguments

`control`	Numeric vector of control group observations
`treatment`	Numeric vector of treatment group observations
`means.only`	Logical; `FALSE` (default) uses full distribution analysis. Set `TRUE` for mean-only comparison
`medians`	Logical; `FALSE` (default) uses means. Set `TRUE` for median-based analysis
`confidence.interval`	Numeric [0,1]; confidence level for effect size bounds (e.g., 0.95)
`tails`	Character; specifies CI tail(s): "both", "left", or "right"
`pairwise`	logical; `FALSE` (default) Returns pairwise certainty tests when set to `pairwise = TRUE`.
`plot`	Logical; `TRUE` (default) generates distribution plot
`robust`	logical; `FALSE` (default) Generates 100 independent random permutations to test results, and returns / plots 95 percent confidence intervals along with robust central tendency of all results for pairwise analysis only.

Details

The key output is the `Certainty` metric, a calibrated probability in $[0, 1]$ representing the likelihood that the groups being compared are the *same* with respect to the chosen comparison mode:

- If `means.only = TRUE`: `Certainty` is the probability that the group *means* (or medians, if `medians = TRUE`) are the same.
- If `means.only = FALSE`: `Certainty` is the probability that the two *entire distributions* are the same.

This makes `Certainty` the conceptual inverse of a classical p-value. A *low* Certainty (e.g., < 0.10) indicates strong evidence of difference, while a *high* Certainty (e.g., > 0.90) indicates strong evidence of similarity.

Value

Returns a list containing:

- `Control_Statistic`: Mean/median of control group
- `Treatment_Statistic`: Mean/median of treatment group
- `Grand_Statistic`: Grand mean/median
- `Control_CDF`: CDF value at grand statistic (control)
- `Treatment_CDF`: CDF value at grand statistic (treatment)
- `Certainty`: Probability that the groups are the *same* (means-only or full distribution depending on `means.only`).
- `Effect_Size_LB`: Lower bound of treatment effect (if confidence.interval requested)
- `Effect_Size_UB`: Upper bound of treatment effect (if confidence.interval requested)
- `Confidence_Level`: Confidence level used (if confidence.interval requested)

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Viole, F. (2017) "Continuous CDFs and ANOVA with NNS" doi:10.2139/ssrn.3007373

Examples

```
 ## Not run:
### Binary analysis and effect size
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.ANOVA(control = x, treatment = y)

### Two variable analysis with no control variable
```

```
A <- cbind(x, y)
NNS.ANOVA(A)

### Medians test
NNS.ANOVA(A, means.only = TRUE, medians = TRUE)

### Multiple variable analysis with no control variable
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100) ; z <- rnorm(100)
A <- cbind(x, y, z)
NNS.ANOVA(A)

### Different length vectors used in a list
x <- rnorm(30) ; y <- rnorm(40) ; z <- rnorm(50)
A <- list(x, y, z)
NNS.ANOVA(A)

## End(Not run)
```

NNS.ARMA *NNS ARMA*

Description

Autoregressive model incorporating nonlinear regressions of component series.

Usage

```
NNS.ARMA(
  variable,
  h = 1,
  training.set = NULL,
  seasonal.factor = TRUE,
  weights = NULL,
  best.periods = 1,
  modulo = NULL,
  mod.only = TRUE,
  negative.values = FALSE,
  method = "nonlin",
  dynamic = FALSE,
  shrink = FALSE,
  plot = TRUE,
  seasonal.plot = TRUE,
  pred.int = NULL
)
```

Arguments

variable	a numeric vector.
h	integer; 1 (default) Number of periods to forecast.
training.set	numeric; NULL (default) Sets the number of variable observations (variable[1 : training.set]) to monitor performance of forecast over in-sample range.
seasonal.factor	logical or integer(s); TRUE (default) Automatically selects the best seasonal lag from the seasonality test. To use weighted average of all seasonal lags set to (seasonal.factor = FALSE). Otherwise, directly input known frequency integer lag to use, i.e. (seasonal.factor = 12) for monthly data. Multiple frequency integers can also be used, i.e. (seasonal.factor = c(12, 24, 36))
weights	numeric or "equal"; NULL (default) sets the weights of the seasonal.factor vector when specified as integers. If (weights = NULL) each seasonal.factor is weighted on its NNS.seas result and number of observations it contains, else an "equal" weight is used.
best.periods	integer; [2] (default) used in conjunction with (seasonal.factor = FALSE), uses the best.periods number of detected seasonal lags instead of ALL lags when (seasonal.factor = FALSE, best.periods = NULL).
modulo	integer(s); NULL (default) Used to find the nearest multiple(s) in the reported seasonal period.
mod.only	logical; TRUE (default) Limits the number of seasonal periods returned to the specified modulo.
negative.values	logical; FALSE (default) If the variable can be negative, set to (negative.values = TRUE). If there are negative values within the variable, negative.values will automatically be detected.
method	options: ("lin", "nonlin", "both", "means"); "nonlin" (default) To select the regression type of the component series, select (method = "both") where both linear and nonlinear estimates are generated. To use a nonlinear regression, set to (method = "nonlin"); to use a linear regression set to (method = "lin"). Means for each subset are returned with (method = "means").
dynamic	logical; FALSE (default) To update the seasonal factor with each forecast point, set to (dynamic = TRUE). The default is (dynamic = FALSE) to retain the original seasonal factor from the inputted variable for all ensuing h.
shrink	logical; FALSE (default) Ensembles forecasts with method = "means".
plot	logical; TRUE (default) Returns the plot of all periods exhibiting seasonality and the variable level reference in upper panel. Lower panel returns original data and forecast.
seasonal.plot	logical; TRUE (default) Adds the seasonality plot above the forecast. Will be set to FALSE if no seasonality is detected or seasonal.factor is set to an integer value.
pred.int	numeric [0, 1]; NULL (default) Plots and returns the associated prediction intervals for the final estimate. Constructed using the maximum entropy bootstrap NNS.meboot on the final estimates.

Value

Returns a vector of forecasts of length (h) if no pred.int specified. Else, returns a data.table with the forecasts as well as lower and upper prediction intervals per forecast point.

Note

For monthly data series, increased accuracy may be realized from forcing seasonal factors to multiples of 12. For example, if the best periods reported are: {37, 47, 71, 73} use (seasonal.factor = c(36, 48, 72)).

(seasonal.factor = FALSE) can be a very computationally expensive exercise due to the number of seasonal periods detected.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Viole, F. (2019) "Forecasting Using NNS" doi:10.2139/ssrn.3382300

Examples

```
## Nonlinear NNS.ARMA using AirPassengers monthly data and 12 period lag
## Not run:
NNS.ARMA(AirPassengers, h = 45, training.set = 100, seasonal.factor = 12, method = "nonlin")

## Linear NNS.ARMA using AirPassengers monthly data and 12, 24, and 36 period lags
NNS.ARMA(AirPassengers, h = 45, training.set = 120, seasonal.factor = c(12, 24, 36), method = "lin")

## Nonlinear NNS.ARMA using AirPassengers monthly data and 2 best periods lag
NNS.ARMA(AirPassengers, h = 45, training.set = 120, seasonal.factor = FALSE, best.periods = 2)

## End(Not run)
```

NNS.ARMA.optim *NNS ARMA Optimizer*

Description

Wrapper function for optimizing any combination of a given seasonal.factor vector in NNS.ARMA. Minimum sum of squared errors (forecast-actual) is used to determine optimum across all NNS.ARMA methods.

Usage

```
NNS.ARMA.optim(
  variable,
  h = NULL,
  training.set = NULL,
  seasonal.factor,
  lin.only = FALSE,
  negative.values = FALSE,
 obj.fn = expression(mean((predicted - actual)^2)/(NNS::Co.LPM(1, predicted, actual,
   target_x = mean(predicted), target_y = mean(actual)) + NNS::Co.UPM(1, predicted,
     actual, target_x = mean(predicted), target_y = mean(actual)))),
  objective = "min",
  linear.approximation = TRUE,
  ncores = NULL,
  pred.int = 0.95,
  print.trace = TRUE,
  plot = FALSE
)
```

Arguments

variable	a numeric vector.
h	integer; NULL (default) Number of periods to forecast out of sample. If NULL, h = length(variable) - training.set.
training.set	integer; NULL (default) Sets the number of variable observations as the training set. See Note below for recommended uses.
seasonal.factor	integers; Multiple frequency integers considered for NNS.ARMA model, i.e. (seasonal.factor = c(12, 24, 36)).
lin.only	logical; FALSE (default) For fast optimization of the linear regression method. More robust than lin.only = TRUE.
negative.values	logical; FALSE (default) If the variable can be negative, set to (negative.values = TRUE). It will automatically select (negative.values = TRUE) if the minimum value of the variable is negative.
obj.fn	expression; expression(cor(predicted, actual, method = "spearman") / sum((predicted - actual)^2)) (default) Rank correlation / sum of squared errors is the default objective function. Any expression(...) using the specific terms predicted and actual can be used.
objective	options: ("min", "max") "max" (default) Select whether to minimize or maximize the objective function obj.fn.
linear.approximation	logical; TRUE (default) Uses the best linear output from NNS.reg to generate a nonlinear and mixture regression for comparison. FALSE is a more exhaustive search over the objective space.

ncores	integer; value specifying the number of cores to be used in the parallelized procedure. If NULL (default), the number of cores to be used is equal to the number of cores of the machine - 1.
pred.int	numeric [0, 1]; 0.95 (default) Returns the associated prediction intervals for the final estimate. Constructed using the maximum entropy bootstrap NNS.meboot on the final estimates.
print.trace	logical; TRUE (default) Prints current iteration information. Suggested as backup in case of error, best parameters to that point still known and copyable!
plot	logical; FALSE (default)

Value

Returns a list containing:

- $period a vector of optimal seasonal periods
- $weights the optimal weights of each seasonal period between an equal weight or NULL weighting
- $obj.fn the objective function value
- $method the method identifying which NNS.ARMA method was used.
- $shrink whether to use the shrink parameter in NNS.ARMA.
- $nns.regress whether to smooth the variable via NNS.reg before forecasting.
- $bias.shift a numerical result of the overall bias of the optimum objective function result. To be added to the final result when using the NNS.ARMA with the derived parameters.
- $errors a vector of model errors from internal calibration.
- $results a vector of length h.
- $lower.pred.int a vector of lower prediction intervals per forecast point.
- $upper.pred.int a vector of upper prediction intervals per forecast point.

Note

- Typically, (training.set = 0.8 * length(variable)) is used for optimization. Smaller samples could use (training.set = 0.9 * length(variable)) (or larger) in order to preserve information.
- The number of combinations will grow prohibitively large, they should be kept as small as possible. seasonal.factor containing an element too large will result in an error. Please reduce the maximum seasonal.factor.
- Set (ncores = 1) if routine is used within a parallel architecture.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Nonlinear NNS.ARMA period optimization using 2 yearly lags on AirPassengers monthly data
## Not run:
nns.optims <- NNS.ARMA.optim(AirPassengers[1:132], training.set = 120,
seasonal.factor = seq(12, 24, 6))

## To predict out of sample using best parameters:
NNS.ARMA.optim(AirPassengers[1:132], h = 12, seasonal.factor = seq(12, 24, 6))

## Incorporate any objective function from external packages (such as \code{Metrics::mape})
NNS.ARMA.optim(AirPassengers[1:132], h = 12, seasonal.factor = seq(12, 24, 6),
obj.fn = expression(Metrics::mape(actual, predicted)), objective = "min")

## End(Not run)
```

NNS.boost *NNS Boost*

Description

Ensemble method for classification using the NNS multivariate regression NNS.reg as the base learner instead of trees.

Usage

```
NNS.boost(
  IVs.train,
  DV.train,
  IVs.test = NULL,
  type = NULL,
  depth = NULL,
  learner.trials = 100,
  epochs = NULL,
  CV.size = NULL,
  balance = FALSE,
  ts.test = NULL,
  threshold = NULL,
  obj.fn = expression(sum((predicted - actual)^2)),
  objective = "min",
  extreme = FALSE,
  features.only = FALSE,
  feature.importance = TRUE,
  pred.int = NULL,
  status = TRUE
)
```

Arguments

IVs.train	a matrix or data frame of variables of numeric or factor data types.
DV.train	a numeric or factor vector with compatible dimensions to (IVs.train).
IVs.test	a matrix or data frame of variables of numeric or factor data types with compatible dimensions to (IVs.train). If NULL, will use (IVs.train) as default.
type	NULL (default). To perform a classification of discrete integer classes from factor target variable (DV.train) with a base category of 1, set to (type = "CLASS"), else for continuous (DV.train) set to (type = NULL).
depth	options: (integer, NULL, "max"); (depth = NULL)(default) Specifies the order parameter in the NNS.reg routine, assigning a number of splits in the regressors, analogous to tree depth.
learner.trials	integer; 100 (default) Sets the number of trials to obtain an accuracy threshold level. If the number of all possible feature combinations is less than selected value, the minimum of the two values will be used.
epochs	integer; 2*length(DV.train) (default) Total number of feature combinations to run.
CV.size	numeric [0, 1]; NULL (default) Sets the cross-validation size. Defaults to a random value between 0.2 and 0.33 for a random sampling of the training set.
balance	logical; FALSE (default) Uses both up and down sampling to balance the classes. type="CLASS" required.
ts.test	integer; NULL (default) Sets the length of the test set for time-series data; typically 2*h parameter value from NNS.ARMA or double known periods to forecast.
threshold	numeric; NULL (default) Sets the obj.fn threshold to keep feature combinations.
obj.fn	expression; expression(sum((predicted - actual)^2)) (default) Sum of squared errors is the default objective function. Any expression(...) using the specific terms predicted and actual can be used. Automatically selects an accuracy measure when (type = "CLASS").
objective	options: ("min", "max") "max" (default) Select whether to minimize or maximize the objective function obj.fn.
extreme	logical; FALSE (default) Uses the maximum (minimum) threshold obtained from the learner.trials, rather than the upper (lower) quintile level for maximization (minimization) objective.
features.only	logical; FALSE (default) Returns only the final feature loadings along with the final feature frequencies.
feature.importance	logical; TRUE (default) Plots the frequency of features used in the final estimate.
pred.int	numeric [0,1]; NULL (default) Returns the associated prediction intervals for the final estimate.
status	logical; TRUE (default) Prints status update message in console.

Value

Returns a vector of fitted values for the dependent variable test set `$results`, prediction intervals `$pred.int`, and the final feature loadings `$feature.weights`, along with final feature frequencies `$feature.frequency`.

Note

- Like a logistic regression, the `(type = "CLASS")` setting is not necessary for target variable of two classes e.g. [0, 1]. The response variable base category should be 1 for classification problems.
- Incorporate any objective function from external packages (such as `Metrics::mape`) via `NNS.boost(..., obj.fn = expression(Metrics::mape(actual, predicted)), objective = "min")`

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. (2016) "Classification Using NNS Clustering Analysis" doi:10.2139/ssrn.2864711

Examples

```
## Using 'iris' dataset where test set [IVs.test] is 'iris' rows 141:150.
## Not run:
a <- NNS.boost(iris[1:140, 1:4], iris[1:140, 5],
IVs.test = iris[141:150, 1:4],
epochs = 100, learner.trials = 100,
type = "CLASS", depth = NULL, balance = TRUE)

## Test accuracy
mean(a$results == as.numeric(iris[141:150, 5]))

## End(Not run)
```

NNS.caus *NNS Causation*

Description

Returns the causality from observational data between two variables.

Usage

```
NNS.caus(
  x,
  y = NULL,
  factor.2.dummy = FALSE,
  tau = 0,
  plot = FALSE,
  p.value = FALSE,
  nperm = 100L,
  permute = c("y", "x", "both"),
  seed = NULL,
  conf.int = 0.95
)
```

Arguments

x	a numeric vector, matrix or data frame.
y	NULL (default) or a numeric vector with compatible dimensions to x.
factor.2.dummy	logical; FALSE (default) Automatically augments variable matrix with numerical dummy variables based on the levels of factors. Includes dependent variable y.
tau	options: ("cs", "ts", integer); 0 (default) Number of lagged observations to consider (for time series data). Otherwise, set (tau = "cs") for cross-sectional data. (tau = "ts") automatically selects the lag of the time series data, while (tau = [integer]) specifies a time series lag.
plot	logical; FALSE (default) Plots the raw variables, tau normalized, and cross-normalized variables.
p.value	logical; FALSE (default) If TRUE, runs a permutation test to compute empirical p-values for the signed causation from x -> y.
nperm	integer; number of permutations to use when p.value = TRUE. Default 100.
permute	one of "both", "y", or "x"; which variable(s) to shuffle when constructing the null distribution.
seed	optional integer seed for reproducibility of the permutation test.
conf.int	numeric; 0.95 (default) confidence level for the partial-moment based interval computed on the permutation null distribution.

Value

If p.value=FALSE returns the original causation vector of length 3 (directional given/received and net), named either "C(x—>y)" or "C(y—>x)" in the third slot. If p.value=TRUE returns a list with components: * causation: the original causation vector as above. * p.value: a list with empirical two-sided and one-sided p-values (x_causes_y, y_causes_x), the null distribution, the observed signed statistic, and metadata (permute, nperm). If p.value=TRUE for a matrix, the function returns a list with components: * causality: the causality matrix. * lower_CI: matrix of lower confidence bounds (partial-moment based). * upper_CI: matrix of upper confidence bounds (partial-moment based). * p.value: matrix of empirical two-sided p-values.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Not run:
## x causes y...
set.seed(123)
x <- rnorm(1000) ; y <- x ^ 2
NNS.caus(x, y, tau = "cs")

## Causal matrix without per factor causation
NNS.caus(iris, tau = 0)

## Causal matrix with per factor causation
NNS.caus(iris, factor.2.dummy = TRUE, tau = 0)

## End(Not run)
```

NNS.CDF *NNS CDF*

Description

This function generates an empirical CDF using partial moment ratios LPM.ratio, and resulting survival, hazard and cumulative hazard functions.

Usage

```
NNS.CDF(variable, degree = 0, target = NULL, type = "CDF", plot = TRUE)
```

Arguments

variable	a numeric vector or data.frame of >= 2 variables for joint CDF.
degree	numeric; (degree = 0) (default) is frequency, (degree = 1) is area.
target	numeric; NULL (default) Must lie within support of each variable.
type	options("CDF", "survival", "hazard", "cumulative hazard"); "CDF" (default) Selects type of function to return for bi-variate analysis. Multivariate analysis is restricted to "CDF".
plot	logical; plots CDF.

Value

Returns:

- "Function" a data.table containing the observations and resulting CDF of the variable.
- "target.value" value from the target argument.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Viole, F. (2017) "Continuous CDFs and ANOVA with NNS" doi:10.2139/ssrn.3007373

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100)
NNS.CDF(x)

## Empirical CDF (degree = 0)
NNS.CDF(x)

## Continuous CDF (degree = 1)
NNS.CDF(x, 1)

## Joint CDF
x <- rnorm(5000) ; y <- rnorm(5000)
A <- cbind(x,y)

NNS.CDF(A, 0)

## Joint CDF with target
NNS.CDF(A, 0, target = rep(0, ncol(A)))

## End(Not run)
```

NNS.copula *NNS Co-Partial Moments Higher Dimension Dependence*

Description

Determines higher dimension dependence coefficients based on co-partial moment matrices ratios.

Usage

```
NNS.copula(
  X,
  target = NULL,
  continuous = TRUE,
  plot = FALSE,
  independence.overlay = FALSE
)
```

Arguments

X	a numeric matrix or data frame.
target	numeric; Typically the mean of Variable X for classical statistics equivalences, but does not have to be. (Vectorized) (target = NULL) (default) will set the target as the mean of every variable.
continuous	logical; TRUE (default) Generates a continuous measure using degree 1 PM.matrix, while discrete FALSE uses degree 0 PM.matrix.
plot	logical; FALSE (default) Generates a 3d scatter plot with regression points.
independence.overlay	logical; FALSE (default) Creates and overlays independent Co.LPM and Co.UPM regions to visually reference the difference in dependence from the data.frame of variables being analyzed. Under independence, the light green and red shaded areas would be occupied by green and red data points respectively.

Value

Returns a multivariate dependence value [0,1].

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. (2016) "Beyond Correlation: Using the Elements of Variance for Conditional Means and Probabilities" doi:10.2139/ssrn.2745308.

Examples

```
## Not run:
set.seed(123)
x <- rnorm(1000) ; y <- rnorm(1000) ; z <- rnorm(1000)
A <- data.frame(x, y, z)
NNS.copula(A, target = colMeans(A), plot = TRUE, independence.overlay = TRUE)

### Target 0
NNS.copula(A, target = rep(0, ncol(A)), plot = TRUE, independence.overlay = TRUE)

## End(Not run)
```

NNS.dep *NNS Dependence*

Description

Returns the dependence and nonlinear correlation between two variables based on higher order partial moment matrices measured by frequency or area.

Usage

```
NNS.dep(x, y = NULL, asym = FALSE, p.value = FALSE, print.map = FALSE)
```

Arguments

x	a numeric vector, matrix or data frame.
y	NULL (default) or a numeric vector with compatible dimensions to x.
asym	logical; FALSE (default) Allows for asymmetrical dependencies.
p.value	logical; FALSE (default) Generates 100 independent random permutations to test results against and plots 95 percent confidence intervals along with all results.
print.map	logical; FALSE (default) Plots quadrant means, or p-value replicates.

Value

Returns the bi-variate "Correlation" and "Dependence" or correlation / dependence matrix for matrix input.

Note

For asymmetrical (asym = TRUE) matrices, directional dependence is returned as ([column variable] —> [row variable]).

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.dep(x, y)

## Correlation / Dependence Matrix
x <- rnorm(100) ; y <- rnorm(100) ; z <- rnorm(100)
B <- cbind(x, y, z)
NNS.dep(B)

## End(Not run)
```

NNS.diff *NNS Numerical Differentiation*

Description

Determines numerical derivative of a given univariate function using projected secant lines on the y-axis. These projected points infer finite steps h, in the finite step method.

Usage

```
NNS.diff(
  f,
  point,
  h = abs(point) * 0.1 + 0.01,
  tol = 1e-10,
  max.iter = NULL,
  digits = 12,
  print.trace = FALSE,
  plot = FALSE
)
```

Arguments

f	an expression or call or a formula with no lhs.
point	numeric; Point to be evaluated for derivative of a given function f.
h	numeric [0, ...]; Initial step for secant projection. Defaults to (h = abs(point) * 0.1 + 0.01).
tol	numeric; Sets the tolerance for the stopping condition of the inferred h. Defaults to (tol = 1e-10).
max.iter	integer; NULL (default) Maximum number of bisection iterations. NULL sets the limit to 100L. For noisy functions the bisection may stall before tol is reached; max.iter provides a hard upper bound.

digits	numeric; Sets the number of digits specification of the output. Defaults to (digits = 12).
print.trace	logical; FALSE (default) Displays each iteration, lower y-intercept, upper y-intercept and inferred h.
plot	logical; plots range, secant lines and y-intercept convergence.

Value

Returns a matrix of values, intercepts, derivatives, inferred step sizes for multiple methods of estimation.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Not run:
f <- function(x) sin(x) / x
NNS.diff(f, 4.1)

## Noisy function with explicit iteration cap
f_noisy <- function(x) sin(x) + rnorm(1, 0, 0.001)
NNS.diff(f_noisy, 1.0, max.iter = 100)

## End(Not run)
```

NNS.distance *NNS Distance*

Description

Internal kernel function for NNS multivariate regression NNS.reg parallel instances.

Usage

```
NNS.distance(rpm, dist.estimate, k = "all", class = NULL)
```

Arguments

rpm	REGRESSION.POINT.MATRIX from NNS.reg
dist.estimate	Vector to generate distances from.
k	n.best from NNS.reg
class	if classification problem.

Value

Returns sum of weighted distances.

`NNS.FSD` *NNS FSD Test*

Description

Bi-directional test of first degree stochastic dominance using lower partial moments.

Usage

```
NNS.FSD(x, y, type = "discrete", plot = TRUE)
```

Arguments

`x`	a numeric vector.
`y`	a numeric vector.
`type`	options: ("discrete", "continuous"); `"discrete"` (default) selects the type of CDF.
`plot`	logical; `TRUE` (default) plots the FSD test.

Value

Returns one of the following FSD results: `"X FSD Y"`, `"Y FSD X"`, or `"NO FSD EXISTS"`.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2016) "LPM Density Functions for the Computation of the SD Efficient Set." Journal of Mathematical Finance, 6, 105-126. doi:10.4236/jmf.2016.61012.

Viole, F. (2017) "A Note on Stochastic Dominance." doi:10.2139/ssrn.3002675.

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.FSD(x, y)

## End(Not run)
```

NNS.FSD.uni *NNS FSD Test uni-directional*

Description

Uni-directional test of first degree stochastic dominance using lower partial moments used in SD Efficient Set routine.

Usage

```
NNS.FSD.uni(x, y, type = "discrete")
```

Arguments

x	a numeric vector.
y	a numeric vector.
type	options: ("discrete", "continuous"); "discrete" (default) selects the type of CDF.

Value

Returns (1) if "X FSD Y", else (0).

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2016) "LPM Density Functions for the Computation of the SD Efficient Set." Journal of Mathematical Finance, 6, 105-126. doi:10.4236/jmf.2016.61012

Viole, F. (2017) "A Note on Stochastic Dominance." doi:10.2139/ssrn.3002675

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.FSD.uni(x, y)

## End(Not run)
```

NNS.gravity *NNS gravity*

Description

Alternative central tendency measure more robust to outliers.

Usage

```
NNS.gravity(x, discrete = FALSE)
```

Arguments

x	vector of data.
discrete	logical; FALSE (default) for discrete distributions.

Value

Returns a numeric value representing the central tendency of the distribution.

Author(s)

Fred Viole, OVVO Financial Systems

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100)
NNS.gravity(x)

## End(Not run)
```

NNS.MC *NNS Monte Carlo Sampling*

Description

Monte Carlo sampling from the maximum entropy bootstrap routine NNS.meboot, ensuring the replicates are sampled from the full [-1,1] correlation space.

Usage

```
NNS.MC(
  x,
  reps = 30,
  lower_rho = -1,
  upper_rho = 1,
  by = 0.01,
  exp = 1,
  type = "spearman",
  drift = TRUE,
  target_drift = NULL,
  target_drift_scale = NULL,
  xmin = NULL,
  xmax = NULL,
  ...
)
```

Arguments

x	vector of data.
reps	numeric; number of replicates to generate, 30 default.
lower_rho	numeric [-1,1]; .01 default will set the from argument in seq(from, to, by).
upper_rho	numeric [-1,1]; .01 default will set the to argument in seq(from, to, by).
by	numeric; .01 default will set the by argument in seq(-1, 1, step).
exp	numeric; 1 default will exponentially weight maximum rho value if exp > 1. Shrinks values towards upper_rho.
type	options("spearman", "pearson", "NNScor", "NNSdep"); type = "spearman"(default) dependence metric desired.
drift	logical; drift = TRUE (default) preserves the drift of the original series.
target_drift	numerical; target_drift = NULL (default) Specifies the desired drift when drift = TRUE, i.e. a risk-free rate of return.
target_drift_scale	numerical; instead of calculating a target_drift, provide a scalar to the existing drift when drift = TRUE.
xmin	numeric; the lower limit for the left tail.
xmax	numeric; the upper limit for the right tail.
...	possible additional arguments to be passed to NNS.meboot.

Value

- ensemble average observation over all replicates as a vector.
- replicates maximum entropy bootstrap replicates as a list for each rho.

References

Vinod, H.D. and Viole, F. (2020) Arbitrary Spearman's Rank Correlations in Maximum Entropy Bootstrap and Improved Monte Carlo Simulations. doi:10.2139/ssrn.3621614

Examples

```
## Not run:
# To generate a set of MC sampled time-series to AirPassengers
MC_samples <- NNS.MC(AirPassengers, reps = 10, lower_rho = -1, upper_rho = 1, by = .5, xmin = 0)

## End(Not run)
```

NNS.meboot *NNS meboot*

Description

Adapted maximum entropy bootstrap routine from meboot https://cran.r-project.org/package=meboot.

Usage

```
NNS.meboot(
  x,
  reps = 999,
  rho = NULL,
  type = "spearman",
  drift = TRUE,
  target_drift = NULL,
  target_drift_scale = NULL,
  trim = 0.1,
  xmin = NULL,
  xmax = NULL,
  reachbnd = TRUE,
  expand.sd = TRUE,
  force.clt = TRUE,
  scl.adjustment = FALSE,
  sym = FALSE,
  elaps = FALSE,
  digits = 6,
  colsubj,
  coldata,
  coltimes,
  ...
)
```

Arguments

x	vector of data.
reps	numeric; number of replicates to generate.
rho	numeric [-1,1] (vectorized); A rho must be provided, otherwise a blank list will be returned. The dependence target is applied to each individual replicate (every replicate is mixed to the requested dependence with the original series); it is **not** applied to the ensemble. See Note.
type	options("spearman", "pearson", "NNScor", "NNSdep"); type = "spearman"(default) dependence metric desired.
drift	logical; drift = TRUE (default) preserves the drift of the original series.
target_drift	numerical; target_drift = NULL (default) Specifies the desired drift when drift = TRUE, i.e. a risk-free rate of return.
target_drift_scale	numerical; instead of calculating a target_drift, provide a scalar to the existing drift when drift = TRUE.
trim	numeric [0,1]; The mean trimming proportion, defaults to trim = 0.1.
xmin	numeric; the lower limit for the left tail.
xmax	numeric; the upper limit for the right tail.
reachbnd	logical; If TRUE potentially reached bounds (xmin = smallest value - trimmed mean and xmax = largest value + trimmed mean) are given when the random draw happens to be equal to 0 and 1, respectively.
expand.sd	logical; If TRUE the standard deviation in the ensemble is expanded. See expand.sd in meboot::meboot.
force.clt	logical; If TRUE the ensemble is forced to satisfy the central limit theorem. See force.clt in meboot::meboot.
scl.adjustment	logical; If TRUE scale adjustment is performed to ensure that the population variance of the transformed series equals the variance of the data.
sym	logical; If TRUE an adjustment is performed to ensure that the ME density is symmetric.
elaps	logical; If TRUE elapsed time during computations is displayed.
digits	integer; 6 (default) number of digits to round output to.
colsubj	numeric; the column in x that contains the individual index. It is ignored if the input data x is not a pdata.frame object.
coldata	numeric; the column in x that contains the data of the variable to create the ensemble. It is ignored if the input data x is not a pdata.frame object.
coltimes	numeric; an optional argument indicating the column that contains the times at which the observations for each individual are observed. It is ignored if the input data x is not a pdata.frame object.
...	possible argument fiv to be passed to expand.sd.

Value

Returns the following row names in a matrix:

- x original data provided as input.
- replicates maximum entropy bootstrap replicates.
- ensemble average observation over all replicates. Being a per-observation mean it is a central summary, not a single series carrying the target dependence; a rank or linear correlation taken directly on the `ensemble` tends to read higher than `rho` (averaging amplifies the shared order), so assess `rho` on the `replicates`.
- xx sorted order stats (xx[1] is minimum value).
- z class intervals limits.
- dv deviations of consecutive data values.
- dvtrim trimmed mean of dv.
- xmin data minimum for ensemble=xx[1]-dvtrim.
- xmax data x maximum for ensemble=xx[n]+dvtrim.
- desintxb desired interval means.
- ordxx ordered x values.
- kappa scale adjustment to the variance of ME density.
- elaps elapsed time.

Note

Vectorized `rho` and `drift` parameters will not vectorize both simultaneously. Also, do not specify `target_drift = NULL`.

The `rho` dependence alignment is calibrated on each individual `replicate`: every replicate is mixed so that its dependence on the original series matches `rho`, in the metric implied by `type`. Assess a result in that **same** metric – a `"NNSdep"` target with NNS.dep`$Dependence` (unsigned) and a `"spearman"`/`"pearson"` target with rank/linear correlation – because a signed correlation taken on an unsigned `"NNSdep"` target is not comparable to `rho` (it can even read negative while the dependence target is met). Separately, the `ensemble` is the per-observation mean of the replicates – a central summary, not a single series carrying the target dependence – so a rank or linear correlation computed directly on the `ensemble` tends to read higher than the per-replicate `rho` (averaging amplifies the shared order). Verify `rho` on the `replicates`, in the metric implied by `type`, rather than on the `ensemble`.

References

- Vinod, H.D. and Viole, F. (2020) Arbitrary Spearman's Rank Correlations in Maximum Entropy Bootstrap and Improved Monte Carlo Simulations. doi:10.2139/ssrn.3621614
- Vinod, H.D. (2013), Maximum Entropy Bootstrap Algorithm Enhancements. doi:10.2139/ssrn.2285041
- Vinod, H.D. (2006), Maximum Entropy Ensembles for Time Series Inference in Economics, *Journal of Asian Economics*, **17**(6), pp. 955-978.
- Vinod, H.D. (2004), Ranking mutual funds using unconventional utility theory and stochastic dominance, *Journal of Empirical Finance*, **11**(3), pp. 353-377.

Examples

```
## Not run:
# To generate an orthogonal rank correlated time-series to AirPassengers
boots <- NNS.meboot(AirPassengers, reps = 100, rho = 0, xmin = 0)

# Verify correlation of replicates ensemble to original
cor(boots["ensemble",]$ensemble, AirPassengers, method = "spearman")

# Plot all replicates
matplot(boots["replicates",]$replicates , type = 'l')

# Plot ensemble
lines(boots["ensemble",]$ensemble, lwd = 3)

# Plot original
lines(1:length(AirPassengers), AirPassengers, lwd = 3, col = "red")

### Vectorized drift with a single rho
boots <- NNS.meboot(AirPassengers, reps = 10, rho = 0, xmin = 0, target_drift = c(1,7))
matplot(do.call(cbind, boots["replicates", ]), type = "l")
lines(1:length(AirPassengers), AirPassengers, lwd = 3, col = "red")

### Vectorized rho with a single target drift
boots <- NNS.meboot(AirPassengers, reps = 10, rho = c(0, .5, 1), xmin = 0, target_drift = 3)
matplot(do.call(cbind, boots["replicates", ]), type = "l")
lines(1:length(AirPassengers), AirPassengers, lwd = 3, col = "red")

### Vectorized rho with a single target drift scale
boots <- NNS.meboot(AirPassengers, reps = 10, rho = c(0, .5, 1), xmin = 0, target_drift_scale = 0.5)
matplot(do.call(cbind, boots["replicates", ]), type = "l")
lines(1:length(AirPassengers), AirPassengers, lwd = 3, col = "red")

## End(Not run)
```

NNS.mode *NNS mode*

Description

Mode of a distribution, either continuous or discrete.

Usage

```
NNS.mode(x, discrete = FALSE, multi = TRUE)
```

Arguments

x	vector of data.
discrete	logical; FALSE (default) for discrete distributions.
multi	logical; TRUE (default) returns multiple mode values.

Value

Returns a numeric value representing the mode of the distribution.

Author(s)

Fred Viole, OVVO Financial Systems

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100)
NNS.mode(x)

## End(Not run)
```

`NNS.moments` *NNS moments*

Description

This function returns the first 4 moments of the distribution.

Usage

```
NNS.moments(x, population = TRUE)
```

Arguments

`x`	a numeric vector.
`population`	logical; `TRUE` (default) Performs the population adjustment. Otherwise returns the sample statistic.

Value

Returns:

- `"$mean"` mean of the distribution.
- `"$variance"` variance of the distribution.
- `"$skewness"` skewness of the distribution.
- `"$kurtosis"` excess kurtosis.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100)
NNS.moments(x)

## End(Not run)
```

`NNS.norm` *NNS Normalization*

Description

Normalizes a matrix of variables based on nonlinear scaling normalization method.

Usage

```
NNS.norm(X, linear = FALSE, chart.type = NULL, location = "topleft")
```

Arguments

`X`	a numeric matrix or data frame, or a list.
`linear`	logical; `FALSE` (default) Performs a linear scaling normalization, resulting in equal means for all variables.
`chart.type`	options: ("l", "b"); `NULL` (default). Set (`chart.type = "l"`) for line, (`chart.type = "b"`) for boxplot.
`location`	Sets the legend location within the plot, per the `x` and `y` co-ordinates used in base graphics legend.

Value

Returns a data.frame of normalized values.

Note

Unequal vectors provided in a list will only generate `linear=TRUE` normalized values.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
A <- cbind(x, y)
NNS.norm(A)

### Normalize list of unequal vector lengths

vec1 <- c(1, 2, 3, 4, 5, 6, 7)
vec2 <- c(10, 20, 30, 40, 50, 60)
vec3 <- c(0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3)

vec_list <- list(vec1, vec2, vec3)
NNS.norm(vec_list)

## End(Not run)
```

NNS.part *NNS Partition Map*

Description

Creates partitions based on partial moment quadrant centroids, iteratively assigning identifications to observations based on those quadrants (unsupervised partitional and hierarchical clustering method). Basis for correlation, dependence NNS.dep, regression NNS.reg routines.

Usage

```
NNS.part(
  x,
  y,
  Voronoi = FALSE,
  type = NULL,
  order = NULL,
  obs.req = 8,
  min.obs.stop = TRUE,
  noise.reduction = "off"
)
```

Arguments

x	a numeric vector.
y	a numeric vector with compatible dimensions to x.
Voronoi	logical; FALSE (default) Displays a Voronoi type diagram using partial moment quadrants.
type	NULL (default) Controls the partitioning basis. Set to (type = "XONLY") for X-axis based partitioning. Defaults to NULL for both X and Y-axis partitioning.
order	integer; Number of partial moment quadrants to be generated. (order = "max") will institute a perfect fit.
obs.req	integer; (8 default) Required observations per cluster where quadrants will not be further partitioned if observations are not greater than the entered value. Reduces minimum number of necessary observations in a quadrant to 1 when (obs.req = 1).
min.obs.stop	logical; TRUE (default) Stopping condition where quadrants will not be further partitioned if a single cluster contains less than the entered value of obs.req.
noise.reduction	the method of determining regression points options for the dependent variable y: ("mean", "median", "mode", "off"); (noise.reduction = "mean") uses means for partitions. (noise.reduction = "median") uses medians instead of means for partitions, while (noise.reduction = "mode") uses modes instead of means for partitions. Defaults to (noise.reduction = "off") where an overall central tendency measure is used, which is the default for the independent variable x.

Value

Returns:

- "dt" a data.frame of x and y observations with their partition assignment "quadrant" in the 3rd column and their prior partition assignment "prior.quadrant" in the 4th column.
- "regression.points" the data.frame of regression points for that given (order = ...).
- "order" the order of the final partition given "min.obs.stop" stopping condition.

Note

min.obs.stop = FALSE will not generate regression points due to unequal partitioning of quadrants from individual cluster observations.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.part(x, y)

## Data.frame of observations and partitions
NNS.part(x, y, order = 1)$dt

## Regression points
NNS.part(x, y, order = 1)$regression.points

## Voronoi style plot
NNS.part(x, y, Voronoi = TRUE)

## Examine final counts by quadrant
DT <- NNS.part(x, y)$dt
DT$counts <- ave(DT$quadrant, DT$quadrant, FUN = length)
DT

## End(Not run)
```

NNS.reg *NNS Regression*

Description

Generates a nonlinear regression based on partial moment quadrant means.

Usage

```
NNS.reg(
  x,
  y,
  factor.2.dummy = TRUE,
  order = NULL,
  dim.red.method = NULL,
  tau = NULL,
  type = NULL,
  point.est = NULL,
  location = "top",
  return.values = TRUE,
  plot = TRUE,
  plot.regions = FALSE,
  residual.plot = TRUE,
  confidence.interval = NULL,
  threshold = 0,
  n.best = NULL,
```

```
  smooth = FALSE,
  noise.reduction = "off",
  dist = "L2",
  ncores = NULL,
  point.only = FALSE,
  multivariate.call = FALSE
)
```

Arguments

x	a vector, matrix or data frame of variables of numeric or factor data types.
y	a numeric or factor vector with compatible dimensions to x.
factor.2.dummy	logical; TRUE (default) Automatically augments variable matrix with numerical dummy variables based on the levels of factors.
order	integer; Controls the number of partial moment quadrant means. Users are encouraged to try different (order = ...) integer settings with (noise.reduction = "off"). (order = "max") will force a limit condition perfect fit.
dim.red.method	options: ("cor", "NNS.dep", "NNS.caus", "all", "equal", numeric vector, NULL) method for determining synthetic X* coefficients (per Dana and Dawes (2004)). Selection of a method automatically engages the dimension reduction regression. The default is NULL for full multivariate regression. (dim.red.method = "NNS.dep") uses NNS.dep for nonlinear dependence weights, while (dim.red.method = "NNS.caus") uses NNS.caus for causal weights. (dim.red.method = "cor") uses standard linear correlation for weights. (dim.red.method = "all") averages all methods for further feature engineering. (dim.red.method = "equal") uses unit weights. Alternatively, user can specify a numeric vector of coefficients.
tau	options("ts", NULL); NULL(default) To be used in conjunction with (dim.red.method = "NNS.caus") or (dim.red.method = "all"). If the regression is using time-series data, set (tau = "ts") for more accurate causal analysis.
type	NULL (default). To perform a classification, set to (type = "CLASS"). Like a logistic regression, it is not necessary for target variable of two classes e.g. [0, 1].
point.est	a numeric or factor vector with compatible dimensions to x. Returns the fitted value y.hat for any value of x.
location	Sets the legend location within the plot, per the x and y co-ordinates used in base graphics legend.
return.values	logical; TRUE (default), set to FALSE in order to only display a regression plot and call values as needed.
plot	logical; TRUE (default) To plot regression.
plot.regions	logical; FALSE (default). Generates 3d regions associated with each regression point for multivariate regressions. Note, adds significant time to routine.
residual.plot	logical; TRUE (default) To plot y.hat and Y.

confidence.interval
: numeric [0, 1]; NULL (default) Plots the associated confidence interval with the estimate and reports the standard error for each individual segment. Also applies the same level for the prediction intervals.

threshold
: numeric [0, 1]; (threshold = 0) (default) Sets the threshold for dimension reduction of independent variables when (dim.red.method) is not NULL.

n.best
: integer; NULL (default) Sets the number of nearest regression points to use in weighting for multivariate regression at sqrt(# of regressors). (n.best = "all") will select and weight all generated regression points. Analogous to k in a k Nearest Neighbors algorithm. Different values of n.best are tested using cross-validation in NNS.stack.

smooth
: logical; FALSE (default) Applies a smoothing spline instead of local linear fit to regression points.

noise.reduction
: the method of determining regression points options: ("mean", "median", "mode", "off"); In low signal:noise situations,(noise.reduction = "mean") uses means for NNS.dep restricted partitions, (noise.reduction = "median") uses medians instead of means for NNS.dep restricted partitions, while (noise.reduction = "mode") uses modes instead of means for NNS.dep restricted partitions. (noise.reduction = "off") uses an overall central tendency measure for partitions.

dist
: options:("L1", "L2", "FACTOR") the method of distance calculation; Selects the distance calculation used. dist = "L2" (default) selects the Euclidean distance and (dist = "L1") selects the Manhattan distance; (dist = "FACTOR") uses a frequency.

ncores
: integer; value specifying the number of cores to be used in the parallelized procedure. If NULL (default), the number of cores to be used is equal to the number of cores of the machine - 1.

point.only
: Internal argument for abbreviated output.

multivariate.call
: Internal argument for multivariate regressions.

Value

UNIVARIATE REGRESSION RETURNS THE FOLLOWING VALUES:

- "R2" provides the goodness of fit;
- "SE" returns the overall standard error of the estimate between y and y.hat;
- "Prediction.Accuracy" returns the correct rounded "Point.est" used in classifications versus the categorical y;
- "derivative" for the coefficient of the x and its applicable range;
- "Point.est" for the predicted value generated;
- "pred.int" lower and upper prediction intervals for the "Point.est" returned using the "confidence.interval" provided;
- "regression.points" provides the points used in the regression equation for the given order of partitions;

- "Fitted.xy" returns a data.frame of x, y, y.hat, resid, NNS.ID, gradient;

MULTIVARIATE REGRESSION RETURNS THE FOLLOWING VALUES:

- "R2" provides the goodness of fit;
- "equation" returns the numerator of the synthetic X* dimension reduction equation as a data.frame consisting of regressor and its coefficient. Denominator is simply the length of all coefficients > 0, returned in last row of equation data.frame.
- "x.star" returns the synthetic X* as a vector;
- "rhs.partitions" returns the partition points for each regressor x;
- "RPM" provides the Regression Point Matrix, the points for each x used in the regression equation for the given order of partitions;
- "Point.est" returns the predicted value generated;
- "pred.int" lower and upper prediction intervals for the "Point.est" returned using the "confidence.interval" provided;
- "Fitted.xy" returns a data.frame of x,y, y.hat, gradient, and NNS.ID.

Note

- Please ensure point.est is of compatible dimensions to x, error message will ensue if not compatible.
- Like a logistic regression, the (type = "CLASS") setting is not necessary for target variable of two classes e.g. [0, 1]. The response variable base category should be 1 for classification problems.
- For low signal:noise instances, increasing the dimension may yield better results using NNS.stack(cbind(x,x), y, method = 1, ...).

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Vinod, H. and Viole, F. (2017) "Nonparametric Regression Using Clusters" doi:10.1007/s10614-01797135

Vinod, H. and Viole, F. (2018) "Clustering and Curve Fitting by Line Segments" doi:10.20944/preprints201801.0090.v1

Viole, F. (2020) "Partitional Estimation Using Partial Moments" doi:10.2139/ssrn.3592491

Dana, J., and Dawes, R. M. (2004). The Superiority of Simple Alternatives to Regression for Social Science Predictions. Journal of Educational and Behavioral Statistics, 29(3), 317–331.

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.reg(x, y)

## Manual {order} selection
NNS.reg(x, y, order = 2)

## Maximum {order} selection
NNS.reg(x, y, order = "max")

## x-only paritioning (Univariate only)
NNS.reg(x, y, type = "XONLY")

## For Multiple Regression:
x <- cbind(rnorm(100), rnorm(100), rnorm(100)) ; y <- rnorm(100)
NNS.reg(x, y, point.est = c(.25, .5, .75))

## For Multiple Regression based on Synthetic X* (Dimension Reduction):
x <- cbind(rnorm(100), rnorm(100), rnorm(100)) ; y <- rnorm(100)
NNS.reg(x, y, point.est = c(.25, .5, .75), dim.red.method = "cor", ncores = 1)

## IRIS dataset examples:
# Dimension Reduction:
NNS.reg(iris[,1:4], iris[,5], dim.red.method = "cor", order = 5, ncores = 1)

# Dimension Reduction using causal weights:
NNS.reg(iris[,1:4], iris[,5], dim.red.method = "NNS.caus", order = 5, ncores = 1)

# Multiple Regression:
NNS.reg(iris[,1:4], iris[,5], order = 2, noise.reduction = "off")

# Classification:
NNS.reg(iris[,1:4], iris[,5], point.est = iris[1:10, 1:4], type = "CLASS")$Point.est

## To call fitted values:
x <- rnorm(100) ; y <- rnorm(100)
NNS.reg(x, y)$Fitted

## To call partial derivative (univariate regression only):
NNS.reg(x, y)$derivative

## End(Not run)
```

`NNS.rescale` *NNS rescale*

Description

Rescale a vector using either min-max scaling or risk-neutral adjustment.

Usage

```
NNS.rescale(x, a, b, method = "minmax", T = NULL, type = "Terminal")
```

Arguments

x	numeric vector; data to rescale (e.g., terminal prices for risk-neutral method).
a	numeric; defines the scaling target: - For method = "minmax": the lower limit of the output range (e.g., 5 to scale to [5, b]). - For method = "riskneutral": the initial price \(S_0 \) (must be positive, e.g., 100), used to set the target mean.
b	numeric; defines the scaling range or rate: - For method = "minmax": the upper limit of the output range (e.g., 10 to scale to [a, 10]). - For method = "riskneutral": the risk-free rate \(r \) (e.g., 0.05), used with \(T \) to adjust the mean.
method	character; scaling method: "minmax" (default) for min-max scaling, or "riskneutral" for risk-neutral adjustment.
T	numeric; time to maturity in years (required for method = "riskneutral", ignored otherwise; e.g., 1). Default is NULL.
type	character; for method = "riskneutral": "Terminal" (default) or "Discounted" (mean = \(S_0 \)).

Value

Returns a rescaled distribution: - For "minmax": values scaled linearly to the range [a, b]. - For "riskneutral": values scaled multiplicatively to a risk-neutral mean (\(S_0 e^(rT) \) if type = "Terminal", or \(S_0 \) if type = "Discounted").

Author(s)

Fred Viole, OVVO Financial Systems

Examples

```
## Not run:
set.seed(123)
# Min-max scaling: a = lower limit, b = upper limit
x <- rnorm(100)
NNS.rescale(x, a = 5, b = 10, method = "minmax")  # Scales to [5, 10]

# Risk-neutral scaling (Terminal): a = S_0, b = r  # Mean approx 105.13
prices <- 100 * exp(cumsum(rnorm(100, 0.001, 0.02)))
NNS.rescale(prices, a = 100, b = 0.05, method = "riskneutral", T = 1, type = "Terminal")

# Risk-neutral scaling (Discounted): a = S_0, b = r  # Mean approx 100
NNS.rescale(prices, a = 100, b = 0.05, method = "riskneutral", T = 1, type = "Discounted")

## End(Not run)
```

NNS.SD.cluster *NNS SD-based Clustering*

Description

Clusters a set of variables by iteratively extracting Stochastic Dominance (SD)-efficient sets, subject to a minimum cluster size.

Usage

```
NNS.SD.cluster(
  data,
  degree = 1,
  type = "discrete",
  min_cluster = 1,
  dendrogram = FALSE
)
```

Arguments

data	A numeric matrix or data frame of variables to be clustered.
degree	Numeric options: (1, 2, 3). Degree of stochastic dominance test.
type	Character, either "discrete" (default) or "continuous"; specifies the type of CDF.
min_cluster	Integer. The minimum number of elements required for a valid cluster.
dendrogram	Logical; FALSE (default). If TRUE, a dendrogram is produced based on a simple "distance" measure between clusters.

Details

The function applies NNS.SD.efficient.set iteratively, peeling off the SD-efficient set at each step if it meets or exceeds min_cluster in size, until no more subsets can be extracted or all variables are exhausted. Variables in each SD-efficient set form a cluster, with any remaining variables aggregated into the final cluster if it meets the min_cluster threshold.

Value

A list with the following components:

- Clusters: A named list of cluster memberships where each element is the set of variable names belonging to that cluster.
- Dendrogram (optional): If dendrogram = TRUE, an hclust object is also returned.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2016) "LPM Density Functions for the Computation of the SD Efficient Set." Journal of Mathematical Finance, 6, 105-126. doi:10.4236/jmf.2016.61012.

Viole, F. (2017) "A Note on Stochastic Dominance." doi:10.2139/ssrn.3002675

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100)
y <- rnorm(100)
z <- rnorm(100)
A <- cbind(x, y, z)

# Perform SD-based clustering (degree 1), requiring at least 2 elements per cluster
results <- NNS.SD.cluster(data = A, degree = 1, min_cluster = 2)
print(results$Clusters)

# Produce a dendrogram as well
results_with_dendro <- NNS.SD.cluster(data = A, degree = 1, min_cluster = 2, dendrogram = TRUE)

## End(Not run)
```

NNS.SD.efficient.set *NNS SD Efficient Set*

Description

Determines the set of stochastic dominant variables for various degrees.

Usage

```
NNS.SD.efficient.set(x, degree, type = "discrete", status = TRUE)
```

Arguments

x	a numeric matrix or data frame.
degree	numeric options: (1, 2, 3); Degree of stochastic dominance test from (1, 2 or 3).
type	options: ("discrete", "continuous"); "discrete" (default) selects the type of CDF.
status	logical; TRUE (default) Prints status update message in console.

Value

Returns set of stochastic dominant variable names.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2016) "LPM Density Functions for the Computation of the SD Efficient Set." Journal of Mathematical Finance, 6, 105-126. doi:10.4236/jmf.2016.61012.

Viole, F. (2017) "A Note on Stochastic Dominance." doi:10.2139/ssrn.3002675

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y<-rnorm(100) ; z<-rnorm(100)
A <- cbind(x, y, z)
NNS.SD.efficient.set(A, 1)

## End(Not run)
```

NNS.seas *NNS Seasonality Test*

Description

Seasonality test based on the coefficient of variation for the variable and lagged component series. A result of 1 signifies no seasonality present.

Usage

```
NNS.seas(variable, modulo = NULL, mod.only = TRUE, plot = TRUE)
```

Arguments

variable	a numeric vector.
modulo	integer(s); NULL (default) Used to find the nearest multiple(s) in the reported seasonal period.
mod.only	logical; TRUE (default) Limits the number of seasonal periods returned to the specified modulo.
plot	logical; TRUE (default) Returns the plot of all periods exhibiting seasonality and the variable level reference.

Value

Returns a matrix of all periods exhibiting less coefficient of variation than the variable with "all.periods"; and the single period exhibiting the least coefficient of variation versus the variable with "best.period"; as well as a vector of "periods" for easy call into NNS.ARMA.optim. If no seasonality is detected, NNS.seas will return ("No Seasonality Detected").

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100)

## To call strongest period based on coefficient of variation:
NNS.seas(x, plot = FALSE)$best.period

## Using modulos for logical seasonal inference:
NNS.seas(x, modulo = c(2,3,5,7), plot = FALSE)

## End(Not run)
```

NNS.SS *NNS Stochastic Superiority*

Description

Computes stochastic superiority between two numeric vectors as the empirical probability that an observation from x exceeds an observation from y, with optional tie adjustment and optional confidence intervals via maximum entropy bootstrap.

Usage

```
NNS.SS(
  x,
  y,
  confidence.interval = FALSE,
  reps = 999,
  ci = 0.95,
  rho = 1
)
```

Arguments

x	a numeric vector.
y	a numeric vector.

`confidence.interval`
: logical; `FALSE` (default) returns only the empirical stochastic superiority measures. Set to `TRUE` to compute bootstrap confidence intervals for `p_star`.

`reps`
: numeric; number of maximum entropy bootstrap replicates used when `confidence.interval = TRUE`. Default is `999`.

`ci`
: numeric in $(0, 1)$; confidence level used for the bootstrap interval when `confidence.interval = TRUE`. Default is `0.95`.

`rho`
: numeric; dependence target passed to `NNS.meboot`. Default is `1`.

Details

`NNS.SS` returns:

$$P(X > Y),$$

the tie probability

$$P(X = Y),$$

and the tie-adjusted stochastic superiority measure

$$P^* = P(X > Y) + \frac{1}{2}P(X = Y).$$

When `confidence.interval = TRUE`, confidence bounds for `P^*` are computed from `NNS.meboot` bootstrap replicates using `LPM.VaR` and `UPM.VaR` with `degree = 0`.

Missing values are removed from both `x` and `y` using `stats::na.omit`. The empirical estimates are computed via a fast sorted comparison routine rather than explicit pairwise expansion of all `x`-`y` combinations.

For continuous data, `p_tie` will typically be zero, so `p_star` and `p_gt` will be identical up to numerical precision. For discrete data, `p_star` provides the standard tie-adjusted superiority measure.

When `confidence.interval = TRUE`, the interval is constructed from the empirical bootstrap distribution of `p_star`, where $\alpha = 1 - ci$. The lower bound is obtained from `LPM.VaR` evaluated at $\alpha/2$, and the upper bound is obtained from `UPM.VaR` evaluated at $\alpha/2$, both with `degree = 0`.

Value

If `confidence.interval = FALSE`, returns a list containing:

`p_gt` empirical probability that `x > y`.

`p_tie` empirical probability that `x = y`.

`p_star` tie-adjusted stochastic superiority probability.

If `confidence.interval = TRUE`, returns a list containing:

`p_gt` empirical probability that `x > y`.

`p_tie` empirical probability that `x = y`.

`p_star` tie-adjusted stochastic superiority probability.

`lower` lower confidence bound for `p_star`.

`upper` upper confidence bound for `p_star`.

`ci` confidence level used.

`reps` number of bootstrap replicates used.

`boot_vals` bootstrap replicate values of `p_star`.

Note

This function measures stochastic superiority as a pairwise exceedance probability. This is distinct from first-, second-, or third-degree stochastic dominance; see NNS.FSD, NNS.SSD, and NNS.TSD for dominance testing.

Author(s)

Fred Viole, OVVO Financial Systems

References

- Vinod, H.D. and Viole, F. (2020) Arbitrary Spearman's Rank Correlations in Maximum Entropy Bootstrap and Improved Monte Carlo Simulations. doi:10.2139/ssrn.3621614
- Viole, F. and Nawrocki, D. (2013) *Nonlinear Nonparametric Statistics: Using Partial Moments*. ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/.

Examples

```
## Not run:
set.seed(123)
x <- rnorm(200, mean = 0.4, sd = 1)
y <- rnorm(200, mean = 0.0, sd = 1)

# Empirical stochastic superiority
NNS.SS(x, y)

# With confidence intervals
NNS.SS(x, y, confidence.interval = TRUE, reps = 999, ci = 0.95)

# Discrete example with ties
x <- sample(1:5, 100, replace = TRUE)
y <- sample(1:5, 100, replace = TRUE)
NNS.SS(x, y)

## End(Not run)
```

NNS.SSD *NNS SSD Test*

Description

Bi-directional test of second degree stochastic dominance using lower partial moments.

Usage

```
NNS.SSD(x, y, plot = TRUE)
```

Arguments

x	a numeric vector.
y	a numeric vector.
plot	logical; TRUE (default) plots the SSD test.

Value

Returns one of the following SSD results: "X SSD Y", "Y SSD X", or "NO SSD EXISTS".

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2016) "LPM Density Functions for the Computation of the SD Efficient Set." Journal of Mathematical Finance, 6, 105-126. doi:10.4236/jmf.2016.61012.

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.SSD(x, y)

## End(Not run)
```

NNS.SSD.uni *NNS SSD Test uni-directional*

Description

Uni-directional test of second degree stochastic dominance using lower partial moments used in SD Efficient Set routine.

Usage

```
NNS.SSD.uni(x, y)
```

Arguments

x	a numeric vector.
y	a numeric vector.

Value

Returns (1) if "X SSD Y", else (0).

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2016) "LPM Density Functions for the Computation of the SD Efficient Set." Journal of Mathematical Finance, 6, 105-126. doi:10.4236/jmf.2016.61012.

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.SSD.uni(x, y)

## End(Not run)
```

NNS.stack *NNS Stack*

Description

Prediction model using the predictions of the NNS base models NNS.reg as features (i.e. meta-features) for the stacked model.

Usage

```
NNS.stack(
  IVs.train,
  DV.train,
  IVs.test = NULL,
  type = NULL,
  obj.fn = expression(sum((predicted - actual)^2)),
  objective = "min",
  optimize.threshold = TRUE,
  dist = "L2",
  CV.size = NULL,
  balance = FALSE,
  ts.test = NULL,
  folds = 5,
  order = NULL,
  method = c(1, 2),
  stack = TRUE,
  dim.red.method = "cor",
  pred.int = NULL,
  status = TRUE,
  ncores = NULL
)
```

Arguments

IVs.train	a vector, matrix or data frame of variables of numeric or factor data types.
DV.train	a numeric or factor vector with compatible dimensions to (IVs.train).
IVs.test	a vector, matrix or data frame of variables of numeric or factor data types with compatible dimensions to (IVs.train). If NULL, will use (IVs.train) as default.
type	NULL (default). To perform a classification of discrete integer classes from factor target variable (DV.train) with a base category of 1, set to (type = "CLASS"), else for continuous (DV.train) set to (type = NULL). Like a logistic regression, this setting is not necessary for target variable of two classes e.g. [0, 1].
obj.fn	expression; expression(sum((predicted - actual)^2)) (default) Sum of squared errors is the default objective function. Any expression() using the specific terms predicted and actual can be used.
objective	options: ("min", "max") "min" (default) Select whether to minimize or maximize the objective function obj.fn.
optimize.threshold	logical; TRUE (default) Will optimize the probability threshold value for rounding in classification problems. If FALSE, returns 0.5.
dist	options:("L1", "L2", "DTW", "FACTOR") the method of distance calculation; Selects the distance calculation used. dist = "L2" (default) selects the Euclidean distance and (dist = "L1") selects the Manhattan distance; (dist = "DTW") selects the dynamic time warping distance; (dist = "FACTOR") uses a frequency.
CV.size	numeric [0, 1]; NULL (default) Sets the cross-validation size if (IVs.test = NULL). Defaults to a random value between 0.2 and 0.33 for a random sampling of the training set.
balance	logical; FALSE (default) Uses both up and down sampling to balance the classes. type="CLASS" required.
ts.test	integer; NULL (default) Sets the length of the test set for time-series data; typically 2*h parameter value from NNS.ARMA or double known periods to forecast.
folds	integer; folds = 5 (default) Select the number of cross-validation folds.
order	options: (integer, "max", NULL); NULL (default) Sets the order for NNS.reg, where (order = "max") is the k-nearest neighbors equivalent, which is suggested for mixed continuous and discrete (unordered, ordered) data.
method	numeric options: (1, 2); Select the NNS method to include in stack. (method = 1) selects NNS.reg; (method = 2) selects NNS.reg dimension reduction regression. Defaults to method = c(1, 2), which will reduce the dimension first, then find the optimal n.best.
stack	logical; TRUE (default) Uses dimension reduction output in n.best optimization, otherwise performs both analyses independently.
dim.red.method	options: ("cor", "NNS.dep", "NNS.caus", "equal", "all") method for determining synthetic X* coefficients. (dim.red.method = "cor") uses standard linear correlation for weights. (dim.red.method = "NNS.dep") (default) uses NNS.dep

	for nonlinear dependence weights, while (dim.red.method = "NNS.caus") uses NNS.caus for causal weights. (dim.red.method = "all") averages all methods for further feature engineering.
pred.int	numeric [0,1]; NULL (default) Returns the associated prediction intervals with each method.
status	logical; TRUE (default) Prints status update message in console.
ncores	integer; value specifying the number of cores to be used in the parallelized subroutine NNS.reg. If NULL (default), the number of cores to be used is equal to the number of cores of the machine - 1.

Value

Returns a vector of fitted values for the dependent variable test set for all models.

- "NNS.reg.n.best" returns the optimum "n.best" parameter for the NNS.reg multivariate regression. "SSE.reg" returns the SSE for the NNS.reg multivariate regression.
- "OBJfn.reg" returns the obj.fn for the NNS.reg regression.
- "NNS.dim.red.threshold" returns the optimum "threshold" from the NNS.reg dimension reduction regression.
- "OBJfn.dim.red" returns the obj.fn for the NNS.reg dimension reduction regression.
- "probability.threshold" returns the optimum probability threshold for classification, else 0.5 when set to FALSE.
- "reg" returns NNS.reg output.
- "reg.pred.int" returns the prediction intervals for the regression output.
- "dim.red" returns NNS.reg dimension reduction regression output.
- "dim.red.pred.int" returns the prediction intervals for the dimension reduction regression output.
- "stack" returns the output of the stacked model.
- "pred.int" returns the prediction intervals for the stacked model.

Note

- Incorporate any objective function from external packages (such as Metrics::mape) via NNS.stack(..., obj.fn = expression(Metrics::mape(actual, predicted)), objective = "min")
- Like a logistic regression, the (type = "CLASS") setting is not necessary for target variable of two classes e.g. [0, 1]. The response variable base category should be 1 for multiple class problems.
- Missing data should be handled prior as well using na.omit or complete.cases on the full dataset.

If error received:

"Error in is.data.frame(x) : object 'RP' not found"

reduce the CV.size.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. (2016) "Classification Using NNS Clustering Analysis" doi:10.2139/ssrn.2864711

Examples

```
## Using 'iris' dataset where test set [IVs.test] is 'iris' rows 141:150.
## Not run:
NNS.stack(iris[1:140, 1:4], iris[1:140, 5], IVs.test = iris[141:150, 1:4], type = "CLASS",
balance = TRUE)

## Using 'iris' dataset to determine [n.best] and [threshold] with no test set.
NNS.stack(iris[ , 1:4], iris[ , 5], type = "CLASS")

## End(Not run)
```

NNS.TSD *NNS TSD Test*

Description

Bi-directional test of third degree stochastic dominance using lower partial moments.

Usage

```
NNS.TSD(x, y, plot = TRUE)
```

Arguments

x	a numeric vector.
y	a numeric vector.
plot	logical; TRUE (default) plots the TSD test.

Value

Returns one of the following TSD results: "X TSD Y", "Y TSD X", or "NO TSD EXISTS".

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2016) "LPM Density Functions for the Computation of the SD Efficient Set." Journal of Mathematical Finance, 6, 105-126. doi:10.4236/jmf.2016.61012.

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.TSD(x, y)

## End(Not run)
```

`NNS.TSD.uni` *NNS TSD Test uni-directional*

Description

Uni-directional test of third degree stochastic dominance using lower partial moments used in SD Efficient Set routine.

Usage

```
NNS.TSD.uni(x, y)
```

Arguments

x	a numeric vector.
y	a numeric vector.

Value

Returns (1) if "X TSD Y", else (0).

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2016) "LPM Density Functions for the Computation of the SD Efficient Set." Journal of Mathematical Finance, 6, 105-126. doi:10.4236/jmf.2016.61012.

Examples

```
## Not run:
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.TSD.uni(x, y)

## End(Not run)
```

NNS.VAR *NNS VAR*

Description

Nonparametric vector autoregressive model incorporating NNS.ARMA estimates of variables into NNS.reg for a multi-variate time-series forecast.

Usage

```
NNS.VAR(
  variables,
  h,
  tau = 1,
  dim.red.method = "cor",
  naive.weights = TRUE,
 obj.fn = expression(mean((predicted - actual)^2)/(NNS::Co.LPM(1, predicted, actual,
   target_x = mean(predicted), target_y = mean(actual)) + NNS::Co.UPM(1, predicted,
     actual, target_x = mean(predicted), target_y = mean(actual)))),
  objective = "min",
  status = TRUE,
  ncores = NULL,
  nowcast = FALSE
)
```

Arguments

variables	a numeric matrix or data.frame of contemporaneous time-series to forecast.
h	integer; 1 (default) Number of periods to forecast. (h = 0) will return just the interpolated and extrapolated values.
tau	positive integer [> 0]; 1 (default) Number of lagged observations to consider for the time-series data. Vector for single lag for each respective variable or list for multiple lags per each variable.
dim.red.method	options: ("cor", "NNS.dep", "NNS.caus", "all") method for reducing regressors via NNS.stack. (dim.red.method = "cor") (default) uses standard linear correlation for dimension reduction in the lagged variable matrix. (dim.red.method = "NNS.dep") uses NNS.dep for nonlinear dependence weights, while (dim.red.method = "NNS.caus") uses NNS.caus for causal weights. (dim.red.method = "all") averages all methods for further feature engineering.
naive.weights	logical; TRUE (default) Equal weights applied to univariate and multivariate outputs in ensemble. FALSE will apply weights based on the number of relevant variables detected.
obj.fn	expression; expression(mean((predicted - actual)^2)) / (Sum of NNS Co-partial moments) (default) MSE / co-movements is the default objective function. Any expression(...) using the specific terms predicted and actual can be used.

objective	options: ("min", "max") "min" (default) Select whether to minimize or maximize the objective function obj.fn.
status	logical; TRUE (default) Prints status update message in console.
ncores	integer; value specifying the number of cores to be used in the parallelized subroutine NNS.ARMA.optim. If NULL (default), the number of cores to be used is equal to the number of cores of the machine - 1.
nowcast	logical; FALSE (default) internal call for frequency alignment in downstream nowcasting applications.

Value

Returns the following matrices of forecasted variables:

- "interpolated_and_extrapolated" Returns a data.frame of the linear interpolated and NNS.ARMA extrapolated values to replace NA values in the original variables argument. This is required for working with variables containing different frequencies, e.g. where NA would be reported for intra-quarterly data when indexed with monthly periods.
- "relevant_variables" Returns the relevant variables from the dimension reduction step.
- "univariate" Returns the univariate NNS.ARMA forecasts.
- "multivariate" Returns the multi-variate NNS.reg forecasts.
- "ensemble" Returns the ensemble of both "univariate" and "multivariate" forecasts.

Note

- "Error in { : task xx failed -}" should be re-run with NNS.VAR(..., ncores = 1).
- Not recommended for factor variables, even after transformed to numeric. NNS.reg is better suited for factor or binary regressor extrapolation.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Viole, F. (2019) "Multi-variate Time-Series Forecasting: Nonparametric Vector Autoregression Using NNS" doi:10.2139/ssrn.3489550

Viole, F. (2020) "NOWCASTING with NNS" doi:10.2139/ssrn.3589816

Viole, F. (2019) "Forecasting Using NNS" doi:10.2139/ssrn.3382300

Vinod, H. and Viole, F. (2017) "Nonparametric Regression Using Clusters" doi:10.1007/s10614-01797135

Vinod, H. and Viole, F. (2018) "Clustering and Curve Fitting by Line Segments" doi:10.20944/preprints201801.0090.v1

Examples

```
## Not run:
#####################################################
### Standard Nonparametric Vector Autoregression ###
#####################################################

set.seed(123)
x <- rnorm(100) ; y <- rnorm(100) ; z <- rnorm(100)
A <- cbind(x = x, y = y, z = z)

### Using lags 1:4 for each variable
NNS.VAR(A, h = 12, tau = 4, status = TRUE)

### Using lag 1 for variable 1, lag 3 for variable 2 and lag 3 for variable 3
NNS.VAR(A, h = 12, tau = c(1,3,3), status = TRUE)

### Using lags c(1,2,3) for variables 1 and 3, while using lags c(4,5,6) for variable 2
NNS.VAR(A, h = 12, tau = list(c(1,2,3), c(4,5,6), c(1,2,3)), status = TRUE)

### PREDICTION INTERVALS
# Store NNS.VAR output
nns_estimate <- NNS.VAR(A, h = 12, tau = 4, status = TRUE)

# Create bootstrap replicates using NNS.meboot
replicates <- NNS.meboot(nns_estimate$ensemble[,1], rho = seq(-1,1,.25))["replicates",]
replicates <- do.call(cbind, replicates)

# Apply UPM.VaR and LPM.VaR for desired prediction interval...95 percent illustrated
# Tail percentage used in first argument per {LPM.VaR} and {UPM.VaR} functions
lower_CIs <- apply(replicates, 1, function(z) LPM.VaR(0.025, 0, z))
upper_CIs <- apply(replicates, 1, function(z) UPM.VaR(0.025, 0, z))

# View results
cbind(nns_estimate$ensemble[,1], lower_CIs, upper_CIs)


########################################
### NOWCASTING with Mixed Frequencies ###
########################################

library(Quandl)
econ_variables <- Quandl(c("FRED/GDPC1", "FRED/UNRATE", "FRED/CPIAUCSL"),type = 'ts',
                         order = "asc", collapse = "monthly", start_date = "2000-01-01")

### Note the missing values that need to be imputed
head(econ_variables)
tail(econ_variables)


NNS.VAR(econ_variables, h = 12, tau = 12, status = TRUE)

## End(Not run)
```

PM.matrix *Partial Moment Matrix*

Description

Builds a list containing CUPM, DUPM, DLPM, CLPM and the overall covariance matrix.

Usage

```
PM.matrix(LPM_degree, UPM_degree, target, variable, pop_adj, norm = FALSE)
```

Arguments

LPM_degree	numeric; lower partial moment degree (0 = freq, 1 = area).
UPM_degree	numeric; upper partial moment degree (0 = freq, 1 = area).
target	numeric vector; thresholds for each column (defaults to colMeans).
variable	numeric matrix or data.frame.
pop_adj	logical; TRUE adjusts population vs. sample moments.
norm	logical; default FALSE. If TRUE, each quadrant matrix is cell-wise normalized so their sum is 1 at each (i,j).

Details

Partial Moment Matrix

Value

A list: $cupm, $dupm, $dlpm, $clpm, $cov.matrix.

Note

When norm = TRUE, each cell (i,j) of the four quadrant matrices is normalized so that their sum equals 1. In this case, $cov.matrix is computed as $cupm + $clpm - $dupm - $dlpm, yielding a dimensionless, signed dependence measure bounded between -1 and 1. This representation discards magnitude information and is therefore a lossy nonlinear correlation matrix. A higher fidelity nonlinear correlation matrix is available via the NNS.dep function.

Examples

```
set.seed(123)
A <- cbind(rnorm(100), rnorm(100), rnorm(100))

# Uses norm = FALSE by default
PM.matrix(1, 1, target = NULL, variable = A, pop_adj = TRUE)

# Enable normalization
PM.matrix(1, 1, target = NULL, variable = A, pop_adj = TRUE, norm = TRUE)

# Use 0's for targets
PM.matrix(1, 1, target = rep(0, ncol(A)), variable = A, pop_adj = TRUE)

# Use variable medians as targets
PM.matrix(1, 1, target = apply(A, 2, "median"), variable = A, pop_adj = TRUE)
```

UPM *Upper Partial Moment*

Description

This function generates a univariate upper partial moment for any degree or target.

Usage

```
UPM(degree, target, variable, excess_ret = FALSE)
```

Arguments

degree	numeric; (degree = 0) is frequency, (degree = 1) is area.
target	numeric; Set to target = mean(variable) for classical equivalences, but does not have to be. When excess_ret = FALSE, this can be a scalar or a vectorized target for the standard partial moment calculation. When excess_ret = TRUE, it is interpreted element-wise as the benchmark/threshold relative to variable.
variable	a numeric vector. data.frame or list type objects are not permissible.
excess_ret	logical; FALSE (default). If TRUE, switches from the standard vectorized-target partial moment to an element-wise excess-deviation calculation. For UPM, this computes pmax(variable - target, 0) raised to degree and averaged. In this mode, target must have length 1 or the same length as variable.

Value

UPM of variable

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
set.seed(123)
x <- rnorm(100)
UPM(0, mean(x), x)
```

`UPM.ratio` *Upper Partial Moment Ratio*

Description

This function generates a standardized univariate upper partial moment of any non-negative degree for a given target.

Usage

```
UPM.ratio(degree, target, variable)
```

Arguments

`degree`	numeric; degree = 0 gives frequency, degree = 1 gives area.
`target`	numeric vector; threshold(s). Defaults to mean(variable).
`variable`	numeric vector or data-frame column to evaluate.

Value

Numeric vector of standardized upper partial moments.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. & Nawrocki, D. (2013) *Nonlinear Nonparametric Statistics: Using Partial Moments* (ISBN:1490523995)

Examples

```
  set.seed(123)
  x <- rnorm(100)
  UPM.ratio(0, mean(x), x)
## Not run:
  plot3d(x, y, Co.UPM(0, sort(x), sort(y), x, y), . . . )

## End(Not run)
```

UPM.VaR *UPM VaR*

Description

Generates an upside value at risk (VaR) quantile based on the Upper Partial Moment ratio.

Usage

```
UPM.VaR(percentile, degree, x)
```

Arguments

percentile	numeric [0, 1]; The percentile for right-tail VaR.
degree	integer; (degree = 0) for discrete distributions, (degree = 1) for continuous distributions.
x	a numeric vector.

Value

Returns a numeric value representing the point at which "percentile" of the area of x is above.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" (ISBN: 1490523995, 2nd edition: https://ovvo-financial.github.io/NNS/book/)

Examples

```
set.seed(123)
x <- rnorm(100)

## For 5th percentile, right-tail
UPM.VaR(0.05, 0, x)
```

Index