Package ‘NNS’

April 23, 2019

Type Package

Title Nonlinear Nonparametric Statistics

Version 0.3.9.1

Date 2019-04-23

Author Fred Viole

Maintainer Fred Viole <ovvo.financial.systems@gmail.com>

Description Nonlinear nonparametric statistics using partial moments. Partial moments are the elements of variance and asymptotically approximate the area of f(x). These robust statistics provide the basis for nonlinear analysis while retaining linear equivalences. NNS offers: Numerical integration, Numerical differentiation, Clustering, Correlation, Dependence, Causal analysis, ANOVA, Regression, Classification, Seasonality, Autoregressive modeling, Normalization and Stochastic dominance. All routines based on: Viole, F. and Nawrocki, D. (2013), Nonlinear Nonparametric Statistics: Using Partial Moments (ISBN: 1490523995).

License GPL-3

LazyData TRUE

RoxygenNote 6.1.0

Depends R (>= 3.3.0)

Imports data.table, rgl, stringr, plyr

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

`Co.LPM` *Co-Lower Partial Moment (Lower Left Quadrant 4)*

Description

This function generates a co-lower partial moment for between two equal length variables for any degree or target.

Usage

```
Co.LPM(degree.x, degree.y, x, y, target.x = mean(x),
  target.y = mean(y))
```

Arguments

`degree.x`	integer; Degree for variable X. (`degree.x = 0`) is frequency, (`degree.x = 1`) is area.
`degree.y`	integer; Degree for variable Y. (`degree.y = 0`) is frequency, (`degree.y = 1`) is area.
`x`	a numeric vector.
`y`	a numeric vector of equal length to `x`.
`target.x`	numeric; Typically the mean of Variable X for classical statistics equivalences, but does not have to be. (Vectorized)
`target.y`	numeric; Typically the mean of Variable Y for classical statistics equivalences, but does not have to be. (Vectorized)

Value

Co-LPM of two variables

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

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

Co.UPM *Co-Upper Partial Moment (Upper Right Quadrant 1)*

Description

This function generates a co-upper partial moment between two equal length variables for any degree or target.

Usage

```
Co.UPM(degree.x, degree.y, x, y, target.x = mean(x),
  target.y = mean(y))
```

Arguments

degree.x	integer; Degree for variable X. (degree.x = 0) is frequency, (degree.x = 1) is area.
degree.y	integer; Degree for variable Y. (degree.y = 0) is frequency, (degree.y = 1) is area.
x	a numeric vector.
y	a numeric vector of equal length to x.
target.x	numeric; Typically the mean of Variable X for classical statistics equivalences, but does not have to be. (Vectorized)
target.y	numeric; Typically the mean of Variable Y for classical statistics equivalences, but does not have to be. (Vectorized)

Value

Co-UPM of two variables

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

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

D.LPM *Divergent-Lower Partial Moment (Lower Right Quadrant 3)*

Description

This function generates a divergent lower partial moment between two equal length variables for any degree or target.

Usage

```
D.LPM(degree.x, degree.y, x, y, target.x = mean(x), target.y = mean(y))
```

Arguments

`degree.x`	integer; Degree for variable X. (`degree.x = 0`) is frequency, (`degree.x = 1`) is area.
`degree.y`	integer; Degree for variable Y. (`degree.y = 0`) is frequency, (`degree.y = 1`) is area.
`x`	a numeric vector.
`y`	a numeric vector of equal length to `x`.
`target.x`	numeric; Typically the mean of Variable X for classical statistics equivalences, but does not have to be. (Vectorized)
`target.y`	numeric; Typically the mean of Variable Y for classical statistics equivalences, but does not have to be. (Vectorized)

Value

Divergent LPM of two variables

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/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 (Upper Left Quadrant 2)*

Description

This function generates a divergent upper partial moment between two equal length variables for any degree or target.

Usage

```
D.UPM(degree.x, degree.y, x, y, target.x = mean(x), target.y = mean(y))
```

Arguments

degree.x	integer; Degree for variable X. (degree.x = 0) is frequency, (degree.x = 1) is area.
degree.y	integer; Degree for variable Y. (degree.y = 0) is frequency, (degree.y = 1) is area.
x	a numeric vector.
y	a numeric vector of equal length to x.
target.x	numeric; Typically the mean of Variable X for classical statistics equivalences, but does not have to be. (Vectorized)
target.y	numeric; Typically the mean of Variable Y for classical statistics equivalences, but does not have to be. (Vectorized)

Value

Divergent UPM of two variables

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

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

dy.dx *Partial Derivative dy/dx*

Description

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

Usage

```
dy.dx(x, y, order = NULL, stn = 0.99, eval.point = median(x),
  deriv.order = 1, h = 0.05, noise.reduction = "mean",
  deriv.method = "FS")
```

Arguments

x	a numeric vector.
y	a numeric vector.
order	integer; Controls the number of partial moment quadrant means. Defaults to (order = NULL) which generates a more accurate derivative for well specified cases.
stn	numeric [0, 1]; Signal to noise parameter, sets the threshold of NNS.dep which reduces "order" when (order = NULL). Defaults to 0.99 to ensure high dependence for higher "order" and endpoint determination.
eval.point	numeric; x point to be evaluated. Defaults to (eval.point = median(x)). Set to (eval.point = "overall") to find an overall partial derivative estimate.
deriv.order	numeric options: (1, 2); 1 (default) for first derivative. For second derivative estimate of f(x), set (deriv.order = 2).
h	numeric [0, ...]; Percentage step used for finite step method. Defaults to h = .05 representing a 5 percent step from the value of the independent variable.
noise.reduction	the method of determing regression points options: ("mean", "median", "mode", "off"); In low signal to noise situations, (noise.reduction = "median") uses medians instead of means for partitions, while (noise.reduction = "mode") uses modes instead of means for partitions. (noise.reduction = "off") allows for maximum possible fit in NNS.reg. Default setting is (noise.reduction = "mean").
deriv.method	method of derivative estimation, options: ("NNS", "FS"); Determines the partial derivative from the coefficient of the NNS.reg output when (deriv.method = "NNS") or generates a partial derivative using the finite step method (deriv.method = "FS") (Defualt).

Value

Returns the value of the partial derivative estimate for the given order.

Note

If a vector of derivatives is required, ensure (deriv.method = "FS").

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Vinod, H. and Viole, F. (2017) "Nonparametric Regression Using Clusters" https://link.springer.com/article/10.1007/s10614-017-9713-5

Examples

```
x <- seq(0, 2 * pi, pi / 100) ; y <-sin(x)
dy.dx(x, y, eval.point = 1.75)

# Vector of derivatives
## Not run:
dy.dx(x, y, eval.point = c(1.75, 2.5), deriv.method = "FS")
## End(Not run)
```

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

Description

Returns the numerical partial derivate 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 = "median", order = NULL, stn = 0.99,
  h = 0.05, n.best = NULL, mixed = FALSE, plot = FALSE,
  noise.reduction = "mean")
```

Arguments

`x`	a numeric matrix or data frame.
`y`	a numeric vector with compatible dimsensions to `x`.
`wrt`	integer; Selects the regressor to differentiate with respect to.
`eval.points`	numeric or options: ("mean", median", "last"); Regressor points to be evaluated. `(eval.points = "median")` (default) to find partial derivatives at the median of every variable. Set to `(eval.points = "last")` to find partial derivatives at the last value of every variable. Set to `(eval.points="mean")` to find partial derivatives at the mean value of every variable. Set to `(eval.points = "all")` to find partial derivatives at every observation.
`order`	integer; NNS.reg `"order"`, defaults to NULL.
`stn`	numeric [0, 1]; Signal to noise parameter, sets the threshold of NNS.dep which reduces `"order"` when `(order = NULL)`. Defaults to 0.99 to ensure high dependence for higher `"order"` and endpoint determination.

h
: numeric [0, ...]; Percentage step used for finite step method. Defaults to `h = .05` representing a 5 percent step from the value of the regressor.

n.best
: integer; Sets the number of closest regression points to use in estimating finite difference points in NNS.reg. `NULL` (default) Uses `ceiling(sqrt(ncol(x)))`.

mixed
: logical; `FALSE` (default) If mixed derivative is to be evaluated, set `(mixed = TRUE)`. Only for single valued `eval.points`.

plot
: logical; `FALSE` (default) Set to `(plot = TRUE)` to view plot.

noise.reduction
: the method of determing regression points options: ("mean", "median", "mode", "off"); In low signal to noise situations, `(noise.reduction = "median")` uses medians instead of means for partitions, while `(noise.reduction = "mode")` uses modes instead of means for partitions. `(noise.reduction = "off")` allows for maximum possible fit in NNS.reg. Default setting is `(noise.reduction = "mean")`.

Value

Returns:

- `dy.d_(...)$"First Derivative"` the 1st derivative
- `dy.d_(...)$"Second Derivative"` the 2nd derivative
- `dy.d_(...)$"Mixed Derivative"` the mixed derivative (for two independent variables only).

Retuns a vector of partial derivatives when `(eval.points = "all")`.

Note

For known function testing and analysis, regressors should be transformed via expand.grid to fill the dimensions with `(order = "max")`. Example provided below.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

```
set.seed(123) ; x_1 <- runif(100) ; x_2 <- runif(100) ; y <- x_1 ^ 2 * x_2 ^ 2
B = cbind(x_1, x_2)
## To find derivatives of y wrt 1st regressor
dy.d_(B, y, wrt = 1, eval.points = c(.5, .5))

## Known function analysis: [y = a ^ 2 * b ^ 2]
## Not run:
x_1 <- seq(0, 1, .1) ; x_2 <- seq(0, 1, .1)
B = expand.grid(x_1, x_2) ; y <- B[ , 1] ^ 2 * B[ , 2] ^ 2
dy.d_(B, y, wrt = 1, eval.points = c(.5, .5), order = "max")
## 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)
```

Arguments

`degree`	integer; (`degree = 0`) is frequency, (`degree = 1`) is area.
`target`	numeric; Typically set to mean, but does not have to be. (Vectorized)
`variable`	a numeric vector.

Value

LPM of variable

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

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 for any degree or target.

Usage

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

Arguments

`degree`	integer; `(degree = 0)` is frequency, `(degree = 1)` is area.
`target`	numeric; Typically set to mean, but does not have to be. (Vectorized)
`variable`	a numeric vector.

Value

Standardized LPM of variable

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

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

## Continuous CDF
LPM.ratio(1, sort(x), x)


## Joint CDF
## Not run:
x <- rnorm(5000) ; y <- rnorm(5000)
plot3d(x, y, Co.LPM(0, 0, sort(x), sort(y), x, y), col = "blue", xlab = "X", ylab = "Y",
zlab = "Probability", box = FALSE)

## End(Not run)
```

`LPM.VaR` *LPM VaR*

Description

Generates a VaR 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 above.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

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

## For 95th percentile VaR (left-tail)
LPM.VaR(0.95, 0, x)
```

NNS.ANOVA *NNS ANOVA*

Description

Analysis of variance (ANOVA) based on lower partial moment CDFs for multiple variables. Returns a degree of certainty the difference in sample means is zero, not a p-value.

Usage

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

Arguments

control	a numeric vector, matrix or data frame.
treatment	NULL (default) a numeric vector, matrix or data frame.
confidence.interval	numeric [0, 1]; The confidence interval surrounding the control mean when (binary = TRUE). Defaults to (confidence.interval = 0.95).
tails	options: ("Left", "Right", "Both"). tails = "Both"(Default) Selects the tail of the distribution to determine effect size.
pairwise	logical; FALSE (defualt) Returns pairwise certainty tests when set to pairwise = TRUE.
plot	logical; TRUE (default) Returns the boxplot of all variables along with grand mean identification. When (binary = TRUE), returns the boxplot of both variables along with grand mean identification and confidence interval thereof.
binary	logical; TRUE (default) Selects binary analysis between a control and treatment variable.

Value

For `(binary = FALSE)` returns the degree certainty the difference in sample means is zero [0, 1].

For `(binary = TRUE)` returns:

- `"Control Mean"`
- `"Treatment Mean"`
- `"Grand Mean"`
- `"Control CDF"`
- `"Treatment CDF"`
- `"Certainty"` the certainty of the same population statistic
- `"Lower Bound Effect"` and `"Upper Bound Effect"` the effect size of the treatment for the specified confidence interval

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

```
### 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)

### 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)
```

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, best.periods = 2, negative.values = FALSE,
  method = "nonlin", dynamic = FALSE, plot = TRUE,
  seasonal.plot = TRUE, intervals = FALSE)
```

Arguments

`variable` a numeric vector.

`h` integer; 1 (default) Number of periods to forecast.

`training.set` numeric; `NULL` (defualt) 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)`)

`best.periods` integer; [2] (default) used in conjuction with (`seasonal.factor = FALSE`), uses the `best.periods` number of detected seasonal lags instead of `ALL` lags when (`seasonal.factor = FALSE`).

`negative.values` logical; `FALSE` (default) If the variable can be negative, set to (`negative.values = TRUE`).

`method` options: ("lin", "nonlin", "both"); `"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 nonlineaer regression, set to (`method = "nonlin"`); to use a linear regression set to (`method = "lin"`).

`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`.

`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.

`intervals` logical; `FALSE` (default) Plots the surrounding forecasts around the final estimate when (`intervals = TRUE`) and (`seasonal.factor = FALSE`). There are no other forecasts to plot when a single `seasonal.factor` is selected.

Value

Returns a vector of forecasts of length (`h`).

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 comutationally expensive exercise due to the number of seasonal periods detected.

If error encountered when (`seasonal.factor = TRUE`):

`"NaNs produced Error in seq.default(length(variable)+1, 1, -lag[i]) : wrong sign in 'by' argument"`

use the combination of (`seasonal.factor = FALSE, best.periods = 1`).

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

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")
## End(Not run)

## Linear NNS.ARMA using AirPassengers monthly data and 12, 24, and 36 period lags
## Not run:
NNS.ARMA(AirPassengers, h = 45, training.set = 120, seasonal.factor = c(12, 24, 36), method = "lin")
## End(Not run)

## Nonlinear NNS.ARMA using AirPassengers monthly data and 2 best periods lag
## Not run:
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, training.set, seasonal.factor,
  negative.values = FALSE, obj.fn = expression(sum((predicted -
  actual)^2)), objective = "min", depth = 1, print.trace = TRUE)
```

Arguments

`variable`	a numeric vector.
`training.set`	numeric; `NULL` (defualt) Sets the number of variable observations
`seasonal.factor`	integers; Multiple frequency integers considered for NNS.ARMA model, i.e. `(seasonal.factor = c(12, 24, 36))`
`negative.values`	logical; `FALSE` (default) If the variable can be negative, set to `(negative.values = TRUE)`.

`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`.
`depth`	integer; `depth = 1` (default) Sets the level from which further combinations are generated containing only members from prior level's best `seasonal.factors`.
`print.trace`	logical; `TRUE` (defualt) Prints current iteration information. Suggested as backup in case of error, best parameters to that point still known and copyable!

Value

Returns a list containing a vector of optimal seasonal periods `$period`, the minimum objective function value `$obj.fn`, and the `$method` identifying which NNS.ARMA method was used.

Note

The number of combinations will grow prohibitively large, they should be kept to a minimum when `(method = "comb")`.

`seasonal.factor` containing an element too large will result in an error. Please reduce the maximum `seasonal.factor`.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

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

## End(Not run)

## Then use optimal parameters in NNS.ARMA to predict 12 periods in-sample
## Not run:
NNS.ARMA(AirPassengers, h=12, training.set=132, seasonal.factor = nns.optims$periods, method = nns.optims$met

## End(Not run)
```

NNS.caus *NNS Causation*

Description

Returns the causality from observational data between two variables.

Usage

```
NNS.caus(x, y, factor.2.dummy = TRUE, tau, time.series = FALSE,
  plot = FALSE)
```

Arguments

x	a numeric vector, matrix or data frame.
y	NULL (default) or a numeric vector with compatible dimsensions to x.
factor.2.dummy	logical; TRUE (default) Automatically augments variable matrix with numerical dummy variables based on the levels of factors.
tau	options: ("cs", "ts", integer); 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.
time.series	logical; FALSE (default) If analyzing time series data with tau = [integer], select (time.series = TRUE). Not required when (tau = "ts").
plot	logical; FALSE (default) Plots the raw variables, tau normalized, and cross-normalized variables.

Value

Returns the directional causation (x —> y) or (y —> x) and net quantity of association. For causal matrix, directional causation is returned as ([column variable] —> [row variable]). Negative numbers represent causal direction attributed to [row variable].

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

```
## x clearly causes y...
set.seed(123)
x <- rnorm(100) ; y <- x ^ 2
NNS.caus(x, y, tau = "cs")

x <- 1:100 ; y <- x^2
NNS.caus(x, y, tau = "ts", time.series = TRUE)
```

```
## Causal matrix
## Not run:
NNS.caus(data.matrix(iris), tau = 0)

## End(Not run)
```

NNS.cor *NNS Correlation*

Description

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

Usage

```
NNS.cor(x, y = NULL, order = NULL, degree = NULL)
```

Arguments

x	a numeric vector, matrix or data frame.
y	NULL (default) or a numeric vector with compatible dimsensions to x.
order	integer; Controls the level of quadrant partitioning. Defualts to (order = NULL). Errors can generally be rectified by setting (order = 1).
degree	integer; (degree = 0) is frequency based correlations, while (degree = 1) is for area based correlations. Defaults to (degree = 0) for smaller number of observations.

Value

Returns nonlinear correlation coefficient between two variables, or nonlinear correlation matrix for matrix input.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

```
set.seed(123)
## Pairwise Correlation
x <- rnorm(100) ; y <- rnorm(100)
NNS.cor(x, y)

## Correlation Matrix
x <- rnorm(100) ; y <- rnorm(100) ; z<-rnorm(100)
```

```
B <- cbind(x, y, z)
NNS.cor(B)
```

NNS.cor.hd *NNS Co-Partial Moments Higher Dimension Correlation*

Description

Determines higher dimension correlation coefficients based on degree 0 co-partial moments.

Usage

```
NNS.cor.hd(x, plot = FALSE, independence.overlay = FALSE)
```

Arguments

x	a numeric matrix or data frame.
plot	logical; FALSE (default) Generates a 3d scatter plot with regression points using plot3d.
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 multivariate nonlinear correlation coefficient

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

```
set.seed(123)
x <- rnorm(1000) ; y <- rnorm(1000) ; z <- rnorm(1000)
A <- data.frame(x, y, z)
NNS.cor.hd(A, plot = TRUE, independence.overlay = TRUE)
```

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, order = NULL, degree = NULL,
  print.map = FALSE)
```

Arguments

`x`	a numeric vector, matrix or data frame.
`y`	`NULL` (default) or a numeric vector with compatible dimsensions to `x`.
`order`	integer; Controls the level of quadrant partitioning. Defaults to `(order = NULL)`. Errors can generally be rectified by setting `(order = 1)`. Will not partition further if less than 4 observations exist in a quadrant.
`degree`	integer; Defaults to NULL to allow number of observations to be `"degree"` determinant.
`print.map`	logical; `FALSE` (default) Plots quadrant means.

Value

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

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

```
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)
```

NNS.diff *NNS Numerical Differentiation*

Description

Determines numerical derivative of a given 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 = 0.1, tol = 1e-10, print.trace = 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 = 0.1).
tol	numeric; Sets the tolerance for the stopping condition of the inferred h. Defualts to (tol = 1e-10).
print.trace	logical; FALSE (default) Displays each iteration, lower y-intercept, upper y-intercept and inferred h.

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" http://amzn.com/1490523995

Examples

```
f <- function(x) sin(x) / x
NNS.diff(f, 4.1)

g <- function(x) sin(x)
NNS.diff(g, 1)
```

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")
```

Arguments

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

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. http://www.scirp.org/Journal/PaperInformation.aspx?PaperID=63817.

Viole, F. (2017) "A Note on Stochastic Dominance." https://ssrn.com/abstract=3002675.

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.FSD(x, y)
```

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. http://www.scirp.org/Journal/PaperInformation.aspx?PaperID=63817.

Viole, F. (2017) "A Note on Stochastic Dominance." https://ssrn.com/abstract=3002675.

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.FSD.uni(x, y)
```

`NNS.norm` *NNS Normalization*

Description

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

Usage

```
NNS.norm(A, chart.type = NULL, linear = FALSE)
```

Arguments

`A`	a numeric matrix or data frame.
`chart.type`	options: ("l", "b"); `NULL` (default). Set `(chart.type = "l")` for line, `(chart.type = "b")` for boxplot.
`linear`	logical; `FALSE` (default) Performs a linear scaling normalization, resulting in equal means for all variables.

Value

Returns a data.frame of normalized values.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

```
set.seed(123)
x <- rnorm(100) ; y<-rnorm(100)
A <- cbind(x, y)
NNS.norm(A)
```

`NNS.part` *NNS Partition Map*

Description

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

Usage

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

Arguments

`x`	a numeric vector.
`y`	a numeric vector with compatible dimsensions 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.
`max.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 `(max.obs.req = 1)`.
`min.obs.stop`	logical; `FALSE` (default) Stopping condition where quadrants will not be further partitioned if a single cluster contains less than the entered value of `max.obs.req`.
`noise.reduction`	the method of determing regression points options: ("mean", "median", "mode", "off"); `(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 = "mean")`, while `(noise.reduction = "off")` will partition quadrant to a single observation for a given `(order = ...)`.

Value

Returns:

- `"dt"` a data.table 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.table of regression points for that given `(order = ...)`.
- `"order"` the `order` of the final partition given `"min.obs.stop"` stopping condition.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" `http://amzn.com/1490523995`

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.part(x, y)
```

```
## Data.table 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 := .N, by = quadrant]
DT
```

NNS.PDF *NNS PDF*

Description

This function generates an empirical PDF using continuous CDFs from LPM.ratio.

Usage

```
NNS.PDF(variable, degree = 1, target = NULL, bins = NULL,
  plot = TRUE)
```

Arguments

`variable`	a numeric vector.
`degree`	integer; (`degree = 0`) is frequency, (`degree = 1`) (default) is area.
`target`	a numeric range of values [a,b] where a < b. `NULL` (default) uses the `variable` observations.
`bins`	numeric; `NULL` (default) Selects number of observations as default bins.
`plot`	logical; plots PDF.

Value

Returns a data.table containing the intervals used and resulting PDF of the variable.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

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

## Custom target range
NNS.PDF(x, target = c(-5, 5))
```

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, stn = 0.98,
  dim.red.method = NULL, tau = NULL, type = NULL, point.est = NULL,
  location = "top", return.values = TRUE, plot = TRUE,
  plot.regions = FALSE, residual.plot = TRUE, std.errors = FALSE,
  confidence.interval = NULL, threshold = 0, n.best = NULL,
  noise.reduction = "mean", norm = NULL, dist = "L2",
  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 dimsensions 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.
stn	numeric [0, 1]; Signal to noise parameter, sets the threshold of (NNS.dep) which reduces ("order") when (order = NULL). Defaults to 0.98 to ensure high dependence for higher ("order") and endpoint determination.
dim.red.method	options: ("cor", "NNS.cor", "NNS.caus", "all", NULL) method for determining synthetic X* coefficients. Selection of a method automatically engages the dimension reduction regression. The default is NULL for full multivariate regression. (dim.red.method = "NNS.cor") uses NNS.cor for nonlinear correlation 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.
tau	options("ts", NULL); NULL(default) If (dim.red.method = "NNS.caus") or (dim.red.method = "all") and 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").

- `point.est` — a numeric or factor vector with compatible dimsensions 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`.
- `std.errors` — logical; `FALSE` (default) To provide standard errors of each linear segment in the `"Fitted.xy"` output.
- `confidence.interval` — numeric [0, 1]; `NULL` (default) Plots the associated confidence interval with the estimate and reports the standard error for each individual segment.
- `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.
- `noise.reduction` — the method of determing 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")` allows for maximum possible fit with a specific `order`.
- `norm` — `NULL` (default) the method of normalization options: ("NNS", "std"); Normalizes `x` between 0 and 1 for multivariate regression when set to `(norm = "std")`, or normalizes `x` according to NNS.norm when set to `(norm = "NNS")`.
- `dist` — options:("L1", "L2") the method of distance calculation; Selects the distance calculation used. `dist = "L2"` (default) selects the Euclidean distance and `(dist = "L1")` seclects the Manhattan distance.
- `multivariate.call` — Internal parameter 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"` returns the `x` point(s) being evaluated;

- `"Point.est"` for the predicted value generated;
- `"regression.points"` provides the points used in the regression equation for the given order of partitions;
- `"Fitted"` returns a vector containing only the fitted values, `y.hat`;
- `"Fitted.xy"` returns a data.table of `x`,`y`, `y.hat`, and `NNS.ID`;

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.table consisting of regressor and its coefficient. Denominator is simply the length of all coefficients > 0, returned in last row of `equation` data.table.
- `"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;
- `"Fitted"` returns a vector containing only the fitted values, `y.hat`;
- `"Fitted.xy"` returns a data.table 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. Also, upon visual inspection of the data, if a highly periodic variable is observed set `(stn = 0)` or `(order = "max")` to ensure a proper fit.

Identical regressors can be used as long as they do not share the same name. For instance, `NNS.reg(cbind(x, 1 * x), y)` will work as `NNS.reg` is not affected by multicollinearity.

`NNS (>= v.0.3.4)` has repurposed parameter `(type = "CLASS")`. `(type = "CLASS")` is now restricted to signifying a classification analysis for `NNS.reg` while `(dim.red.method)` enables dimension reduction regressions.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Vinod, H. and Viole, F. (2017) "Nonparametric Regression Using Clusters" https://link.springer.com/article/10.1007/s10614-017-9713-5

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.reg(x, y)

## Manual {order} selection
## Not run:
NNS.reg(x, y, order = 2)
```

```
## End(Not run)

## Maximum {order} selection
## Not run:
NNS.reg(x, y, order = "max")
## End(Not run)

## x-only paritioning (Univariate only)
## Not run:
NNS.reg(x, y, type = "XONLY")
## End(Not run)

## For Multiple Regression:
## Not run:
x <- cbind(rnorm(100), rnorm(100), rnorm(100)) ; y <- rnorm(100)
NNS.reg(x, y, point.est = c(.25, .5, .75))
## End(Not run)

## For Multiple Regression based on Synthetic X* (Dimension Reduction):
## Not run:
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")
## End(Not run)

## IRIS dataset examples:
# Dimension Reduction:
## Not run:
NNS.reg(iris[,1:4], iris[,5], dim.red.method = "cor", order = 5)
## End(Not run)

# Dimension Reduction using causal weights:
## Not run:
NNS.reg(iris[,1:4], iris[,5], dim.red.method = "NNS.caus", order = 5)
## End(Not run)

# Multiple Regression:
## Not run:
NNS.reg(iris[,1:4], iris[,5], order = 2, noise.reduction = "off")
## End(Not run)

# Classification:
## Not run:
NNS.reg(iris[,1:4], iris[,5], point.est = iris[1:10, 1:4], type = "CLASS")$Point.est
## End(Not run)

## To call fitted values:
## Not run:
x <- rnorm(100) ; y <- rnorm(100)
NNS.reg(x, y)$Fitted
## End(Not run)

## To call partial derivative (univariate regression only):
## Not run:
NNS.reg(x, y)$derivative
## 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")
```

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.

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. http://www.scirp.org/Journal/PaperInformation.aspx?PaperID=63817.

Viole, F. (2017) "A Note on Stochastic Dominance." https://ssrn.com/abstract=3002675.

Examples

```
set.seed(123)
x <- rnorm(100) ; y<-rnorm(100) ; z<-rnorm(100)
A <- cbind(x, y, z)
NNS.SD.efficient.set(A, 1)
```

NNS.seas *NNS Seasonality Test*

Description

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

Usage

```
NNS.seas(variable, plot = TRUE)
```

Arguments

variable	a numeric vector.
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 variance than the variable with "all.periods"; and the single period exhibiting the least coefficient of variance versus the variable with "best.period". 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" http://amzn.com/1490523995

Examples

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

## To call strongest period based on coefficient of variance:
NNS.seas(x)$best.period
```

NNS.SSD *NNS SSD Test*

Description

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

Usage

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

Arguments

x	a numeric vector.
y	a numeric vector.

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. http://www.scirp.org/Journal/PaperInformation.aspx?PaperID=63817.

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.SSD(x, y)
```

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. http://www.scirp.org/Journal/PaperInformation.aspx?PaperID=63817.

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.SSD.uni(x, y)
```

`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, CV.size = NULL,
  obj.fn = expression(sum((predicted - actual)^2)), objective = "min",
  order = NULL, norm = NULL, method = c(1, 2),
  dim.red.method = "cor", seed = 123)
```

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 dimsensions to `(IVs.train)`.
`IVs.test`	a vector, matrix or data frame of variables of numeric or factor data types.
`CV.size`	numeric [0, 1]; `NULL` (default) Sets the cross-validation size if `(IVs.test = NULL)`. Defaults to 0.25 for a 25 percent random sampling of the training set under `(CV.size = NULL)`.
`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`.
`order`	integer; `NULL` (default) Sets the order for NNS.reg, where `(order = 'max')` is the k-nearest neighbors equivalent.
`norm`	options: ("std", "NNS", NULL); `NULL` (default) 3 settings offered: `NULL`, `"std"`, and `"NNS"`. Selects the `norm` parameter in NNS.reg.
`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)`, including both NNS regression methods in the stack.
`dim.red.method`	options: ("cor", "NNS.cor", "NNS.caus", "all") method for determining synthetic X* coefficients. `(dim.red.method = "cor")` (default) uses standard linear correlation for weights. `(dim.red.method = "NNS.cor")` uses NNS.cor for nonlinear correlation weights, while `(dim.red.method = "NNS.caus")` uses NNS.caus for causal weights. `(dim.red.method = "all")` averages all methods for further feature engineering.
`seed`	numeric; 123 (default) Sets seed for CV sampling.
`weight`	options: ("SSE", "Features") method for selecting model output weight; Set `(weight = "SSE")` for optimum parameters and weighting based on each base model's sum of squared errors. `(weight = "Feautures")` uses a weighting based on the number of features present, whereby logistic NNS.reg receives higher relative weights for more regressors. Defaults to `"SSE"`.

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"` paramater 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.
- `"reg"` returns NNS.reg output.
- `"dim.red"` returns NNS.reg dimension reduction regression output.
- `"stack"` returns the output of the stacked model.

Note

If character variables are used, transform them first to factors using as.factor, or data.matrix to ensure overall dataset is numeric. A multifunction sapply can also be applied to the overall dataset: `data <- sapply(data,function(x){as.factor(x) ; as.numeric(x)})`. Then run `NNS.stack` with transormed variables.

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" https://ssrn.com/abstract=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])
## End(Not run)

 ## Using 'iris' dataset to determine [n.best] and [threshold] with no test set.
 ## Not run:
 NNS.stack(iris[ , 1:4], iris[ , 5])
## End(Not run)

 ## Selecting NNS.reg and dimension reduction techniques.
 ## Not run:
 NNS.stack(iris[1:140, 1:4], iris[1:140, 5], iris[141:150, 1:4], method = c(1, 2))
## End(Not run)

 ## Using classification accuracy in the [obj.fn].
 ## Not run:
 NNS.stack(iris[1:140, 1:4], iris[1:140, 5], iris[141:150, 1:4], method = c(1, 2),  obj.fn = expression( mean(ro
## End(Not run)
```

`NNS.term.matrix` *NNS Term Matrix*

Description

Generates a term matrix for text classification use in NNS.reg.

Usage

```
NNS.term.matrix(x, oos = NULL, names = FALSE)
```

Arguments

`x`	Text A two column dataset should be used. Concatenate text from original sources to comply with format. Also note the possiblity of factors in `"DV"`, so `"as.numeric(as.character(...))"` is used to avoid issues.
`oos`	Out-of-sample text dataset to be classified.
`names`	Column names for `"IV"` and `"oos"`. Defaults to FALSE.

Value

Returns the text as independent variables `"IV"` and the classification as the dependent variable `"DV"`. Out-of-sample independent variables are returned with `"OOS"`.

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

```
x <- data.frame(cbind(c("sunny", "rainy"), c(1, -1)))
NNS.term.matrix(x)

### Concatenate Text with space seperator, cbind with "DV"
x <- data.frame(cbind(c("sunny", "rainy"), c("windy", "cloudy"), c(1, -1)))
x <- data.frame(cbind(paste(x[ , 1], x[ , 2], sep = " "), as.numeric(as.character(x[ , 3]))))
NNS.term.matrix(x)


### NYT Example
## Not run:
require(RTextTools)
data(NYTimes)

### Concatenate Columns 3 and 4 containing text, with column 5 as DV
NYT=data.frame(cbind(paste(NYTimes[ , 3], NYTimes[ , 4], sep = " "),
                     as.numeric(as.character(NYTimes[ , 5]))))
NNS.term.matrix(NYT)
## 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)
```

Arguments

x	a numeric vector.
y	a numeric vector.

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. http://www.scirp.org/Journal/PaperInformation.aspx?PaperID=63817.

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.TSD(x, y)
```

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. http://www.scirp.org/Journal/PaperInformation.aspx?PaperID=63817.

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100)
NNS.TSD.uni(x, y)
```

PM.matrix *Partial Moment Matrix*

Description

This function generates a co-partial moment matrix for the specified co-partial moment.

Usage

```
PM.matrix(LPM.degree, UPM.degree, target, variable, pop.adj = FALSE)
```

Arguments

LPM.degree	integer; Degree for `variable` below `target` deviations. `(degree = 0)` is frequency, `(degree = 1)` is area.
UPM.degree	integer; Degree for `variable` above `target` deviations. `(degree = 0)` is frequency, `(degree = 1)` is area.
target	numeric; Typically the mean of Variable X for classical statistics equivalences, but does not have to be. (Vectorized) `(target = "mean")` will set the target as the mean of every variable.
variable	a numeric matrix or data.frame.
pop.adj	logical; `FALSE` (default) Adjusts the sample co-partial moment matrices for population statistics.

Value

Matrix of partial moment quadrant values. Uncalled quadrants will return a matrix of zeros.

Note

For divergent asymmetrical `"D.LPM" and "D.UPM"` matrices, matrix is `D.LPM(column,row,...)`.

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

```
set.seed(123)
x <- rnorm(100) ; y <- rnorm(100) ; z <- rnorm(100)
A <- cbind(x,y,z)
PM.matrix(LPM.degree = 1, UPM.degree = 1, target = "mean", variable = A)

## Calling Individual Partial Moment Quadrants
cov.mtx = PM.matrix(LPM.degree = 1, UPM.degree = 1, target = "mean", variable = A)
cov.mtx$cupm
```

```
## Full covariance matrix
cov.mtx$matrix
```

UPM *Upper Partial Moment*

Description

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

Usage

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

Arguments

degree	integer; (degree = 0) is frequency, (degree = 1) is area.
target	numeric; Typically set to mean, but does not have to be. (Vectorized)
variable	a numeric vector.

Value

UPM of variable

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

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 for any degree or target.

Usage

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

Arguments

`degree`	integer; (`degree = 0`) is frequency, (`degree = 1`) is area.
`target`	numeric; Typically set to mean, but does not have to be. (Vectorized)
`variable`	a numeric vector.

Value

Standardized UPM of variable

Author(s)

Fred Viole, OVVO Financial Systems

References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" http://amzn.com/1490523995

Examples

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

## Joint Upper CDF
## Not run:
x <- rnorm(5000) ; y <- rnorm(5000)
plot3d(x, y, Co.UPM(0, 0, sort(x), sort(y), x, y), col = "blue", xlab = "X", ylab = "Y",
zlab = "Probability", box = FALSE)

## End(Not run)
```

`UPM.VaR` *UPM VaR*

Description

Generates an upside VaR 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 below.

Examples

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

## For 95th percentile VaR (right-tail)
UPM.VaR(0.95, 0, x)
```

Index