

# Package ‘NNS’

May 24, 2020

**Type** Package

**Title** Nonlinear Nonparametric Statistics

**Version** 0.5.3

**Date** 2020-05-24

**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

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

**LazyData** TRUE

**RoxygenNote** 7.0.2

**Depends** R (>= 3.3.0), doParallel

**Imports** data.table, dtw, meboot, rgl, stringr, tdigest

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

## R topics documented:

Co.LPM . . . . .	2
Co.UPM . . . . .	3
D.LPM . . . . .	4
D.UPM . . . . .	5
dy.dx . . . . .	6
dy.d_ . . . . .	7
LPM . . . . .	9
LPM.ratio . . . . .	9
LPM.VaR . . . . .	10
NNS.ANOVA . . . . .	11
NNS.ARMA . . . . .	13

NNS.ARMA.optim . . . . .	15
NNS.boost . . . . .	17
NNS.caus . . . . .	19
NNS.CDF . . . . .	20
NNS.cor . . . . .	21
NNS.dep . . . . .	22
NNS.dep.base . . . . .	24
NNS.dep.hd . . . . .	25
NNS.diff . . . . .	26
NNS.distance . . . . .	27
NNS.FSD . . . . .	27
NNS.FSD.uni . . . . .	28
NNS.meboot . . . . .	29
NNS.norm . . . . .	31
NNS.part . . . . .	32
NNS.PDF . . . . .	33
NNS.reg . . . . .	34
NNS.SD.efficient.set . . . . .	38
NNS.seas . . . . .	39
NNS.SSD . . . . .	40
NNS.SSD.uni . . . . .	40
NNS.stack . . . . .	41
NNS.term.matrix . . . . .	44
NNS.TSD . . . . .	45
NNS.TSD.uni . . . . .	45
NNS.VAR . . . . .	46
PM.matrix . . . . .	49
UPM . . . . .	50
UPM.ratio . . . . .	51
UPM.VaR . . . . .	52

<b>Index</b>	<b>53</b>
--------------	-----------

---

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.

<code>x</code>	a numeric vector.
<code>y</code>	a numeric vector of equal length to <code>x</code> .
<code>target.x</code>	numeric; Typically the mean of Variable X for classical statistics equivalences, but does not have to be. (Vectorized)
<code>target.y</code>	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"  
<https://www.amazon.com/dp/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**

<code>degree.x</code>	integer; Degree for variable X. ( <code>degree.x = 0</code> ) is frequency, ( <code>degree.x = 1</code> ) is area.
<code>degree.y</code>	integer; Degree for variable Y. ( <code>degree.y = 0</code> ) is frequency, ( <code>degree.y = 1</code> ) is area.
<code>x</code>	a numeric vector.
<code>y</code>	a numeric vector of equal length to <code>x</code> .
<code>target.x</code>	numeric; Typically the mean of Variable X for classical statistics equivalences, but does not have to be. (Vectorized)
<code>target.y</code>	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"  
<https://www.amazon.com/dp/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"  
<https://www.amazon.com/dp/1490523995>

## Examples

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

---

D.UPM	<i>Divergent-Upper Partial Moment (Upper Left Quadrant 2)</i>
-------	---

---

## 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"  
<https://www.amazon.com/dp/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,
  eval.point = median(x),
  deriv.order = 1,
  h = 0.05,
  deriv.method = "FS"
)
```

**Arguments**

x	a numeric vector.
y	a numeric vector.
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.
deriv.method	method of derivative estimation, options: ("NNS", "FS"); Determines the partial derivative from the coefficient of the <a href="#">NNS.reg</a> output when (deriv.method = "NNS") or generates a partial derivative using the finite step method (deriv.method = "FS") (Default).

**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" <https://www.amazon.com/dp/1490523995>

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

## Examples

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

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

---

dy.d_	<i>Partial Derivative <math>dy/d_{[wrt]}</math></i>
-------	---

---

## 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",
  folds = 5,
  mixed = FALSE,
  plot = FALSE,
  ncores = NULL,
  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: ("mean", "median", "last", "all"); Regressor points to be evaluated. (eval.points = "median") (default) to find the average partial derivative at the median of the variable with respect to. Set to (eval.points = "last") to find the average partial derivative at the last observation of the variable with respect to (relevant for time-series data). Set to (eval.points="mean") to find the average partial derivative at the mean of the variable with respect to. Set to (eval.points = "all") to find the average partial derivative at every observation of the variable with respect to.
folds	integer; 5 (default) Sets the number of folds in the <a href="#">NNS.stack</a> procedure for optimal n.best parameter.
mixed	logical; FALSE (default) If mixed derivative is to be evaluated, set (mixed = TRUE).
plot	logical; FALSE (default) Set to (plot = TRUE) to view plot. Default setting is (noise.reduction = "mean").

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.
messages	logical; TRUE (default) Prints status messages of cross-validation on n.best parameter for <a href="#">NNS.reg</a> .

### 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).

### Author(s)

Fred Viole, OVVO Financial Systems

### References

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments"  
<https://www.amazon.com/dp/1490523995>

### Examples

```
## Not run:
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 average partial derivative of y wrt 1st regressor,
only supply 1 value in [eval.points]:
dy.d_(B, y, wrt = 1, eval.points = c(.5))

dy.d_(B, y, wrt = 1, eval.points = mean(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 = "all")
plot(B[,1], apd[[1]])

## To find derivatives of y wrt 1st regressor and specified 2nd regressor
dy.d_(B, y, wrt = 1, eval.points = c(.5, .5))

## Known function analysis: [y = a ^ 2 * b ^ 2]
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))

## End(Not run)
```



---

LPM	<i>Lower Partial Moment</i>
-----	-----------------------------

---

**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"  
<https://www.amazon.com/dp/1490523995>

**Examples**

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

---

LPM.ratio	<i>Lower Partial Moment RATIO</i>
-----------	-----------------------------------

---

**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"  
<https://www.amazon.com/dp/1490523995>

Viole, F. (2017) "Continuous CDFs and ANOVA with NNS" <https://ssrn.com/abstract=3007373>

**Examples**

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

## Not run:
## Empirical CDF (degree = 0)
lpm_cdf <- LPM.ratio(0, sort(x), x)
plot(sort(x), lpm_cdf)

## Continuous CDF (degree = 1)
lpm_cdf_1 <- LPM.ratio(1, sort(x), x)
plot(sort(x), lpm_cdf_1)

## Joint CDF
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 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 (vectorized).
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"  
<https://www.amazon.com/dp/1490523995>

**Examples**

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

## For 5% quantile, left-tail
LPM.VaR(0.05, 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**

<code>control</code>	a numeric vector, matrix or data frame.
<code>treatment</code>	NULL (default) a numeric vector, matrix or data frame.
<code>confidence.interval</code>	numeric [0, 1]; The confidence interval surrounding the control mean when (binary = TRUE). Defaults to (confidence.interval = 0.95).
<code>tails</code>	options: ("Left", "Right", "Both"). tails = "Both"(Default) Selects the tail of the distribution to determine effect size.
<code>pairwise</code>	logical; FALSE (default) Returns pairwise certainty tests when set to pairwise = TRUE.
<code>plot</code>	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.
<code>binary</code>	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"  
<https://www.amazon.com/dp/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,
  weights = NULL,
  best.periods = 2,
  negative.values = FALSE,
  method = "nonlin",
  dynamic = FALSE,
  plot = TRUE,
  seasonal.plot = TRUE,
  conf.intervals = NULL,
  ncores = NULL
)
```

## Arguments

<code>variable</code>	a numeric vector.
<code>h</code>	integer; 1 (default) Number of periods to forecast.
<code>training.set</code>	numeric; NULL (default) Sets the number of variable observations ( <code>variable[1 : training.set]</code> ) to monitor performance of forecast over in-sample range.
<code>seasonal.factor</code>	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 ( <code>seasonal.factor = FALSE</code> ). Otherwise, directly input known frequency integer lag to use, i.e. ( <code>seasonal.factor = 12</code> ) for monthly data. Multiple frequency integers can also be used, i.e. ( <code>seasonal.factor = c(12, 24, 36)</code> )
<code>weights</code>	numeric; NULL (default) sets the weights of the <code>seasonal.factor</code> vector when specified as integers. If ( <code>weights = NULL</code> ) each <code>seasonal.factor</code> is weighted on its <a href="#">NNS.seas</a> result and number of observations it contains.
<code>best.periods</code>	integer; [2] (default) used in conjunction with ( <code>seasonal.factor = FALSE</code> ), uses the <code>best.periods</code> number of detected seasonal lags instead of ALL lags when ( <code>seasonal.factor = FALSE</code> ).

<code>negative.values</code>	logical; FALSE (default) If the variable can be negative, set to ( <code>negative.values = TRUE</code> ). If there are negative values within the variable, <code>negative.values</code> will automatically be detected.
<code>method</code>	options: ("lin", "nonlin", "both"); "nonlin" (default) To select the regression type of the component series, select ( <code>method = "both"</code> ) where both linear and nonlinear estimates are generated. To use a nonlinear regression, set to ( <code>method = "nonlin"</code> ); to use a linear regression set to ( <code>method = "lin"</code> ).
<code>dynamic</code>	logical; FALSE (default) To update the seasonal factor with each forecast point, set to ( <code>dynamic = TRUE</code> ). The default is ( <code>dynamic = FALSE</code> ) to retain the original seasonal factor from the inputted variable for all ensuing h.
<code>plot</code>	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.
<code>seasonal.plot</code>	logical; TRUE (default) Adds the seasonality plot above the forecast. Will be set to FALSE if no seasonality is detected or <code>seasonal.factor</code> is set to an integer value.
<code>conf.intervals</code>	numeric [0, 1]; NULL (default) Plots and returns the associated confidence intervals for the final estimate. Constructed using the maximum entropy bootstrap <a href="#">meboot</a> on the final estimates.
<code>ncores</code>	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 half the number of cores of the machine - 1.

### Value

Returns a vector of forecasts of length (h) if no `conf.intervals` specified. Else, returns a [data.table](#) with the forecasts as well as lower and upper confidence 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.

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" <https://www.amazon.com/dp/1490523995>

Viole, F. (2019) "Forecasting Using NNS" <https://ssrn.com/abstract=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	<i>NNS ARMA Optimizer</i>
----------------	---------------------------

---

## 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",
  linear.approximation = TRUE,
  print.trace = TRUE,
  ncores = NULL
)
```

## Arguments

<code>variable</code>	a numeric vector.
<code>training.set</code>	numeric; Sets the number of variable observations as the training set. See Note below for recommended uses.
<code>seasonal.factor</code>	integers; Multiple frequency integers considered for <a href="#">NNS.ARMA</a> model, i.e. ( <code>seasonal.factor = c(12, 24, 36)</code> )
<code>negative.values</code>	logical; FALSE (default) If the variable can be negative, set to ( <code>negative.values = TRUE</code> ).
<code>obj.fn</code>	expression; <code>expression(sum((predicted - actual)^2))</code> (default) Sum of squared errors is the default objective function. Any <code>expression()</code> using the specific terms <code>predicted</code> and <code>actual</code> can be used.

objective	options: ("min", "max") "min" (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.
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!
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 half the number of cores of the machine.

### 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.
- \$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.

### Note

- Typically, (training.set = length(variable) - 2 \* length(forecast horizon)) is used for optimization. Smaller samples would use (training.set = length(variable) - length(forecast horizon)) 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.
- If variable cannot logically assume negative values, then the \$bias.shift must be limited to 0 via a pmax(0, ...) call.

### Author(s)

Fred Violo, OVVO Financial Systems

### References

Violo, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" <https://www.amazon.com/dp/1490523995>

### 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))

## Then use optimal parameters in NNS.ARMA to predict 12 periods in-sample.
```



```
## Note the {$bias.shift} usage in the {NNS.ARMA} function:
nns.estimates <- NNS.ARMA(AirPassengers, h = 12, training.set = 132,
seasonal.factor = nns.optims$periods, method = nns.optims$method) + nns.optims$bias.shift

## If variable cannot logically assume negative values
nns.estimates <- pmax(0, nns.estimates)

## End(Not run)
```

NNS.boost

*NNS Boost*

## Description

Ensemble method for classification using the predictions of the NNS multivariate regression [NNS.reg](#) collected from uncorrelated feature combinations.

## Usage

```
NNS.boost(
  IVs.train,
  DV.train,
  IVs.test = NULL,
  type = NULL,
  representative.sample = FALSE,
  depth = "max",
  n.best = NULL,
  learner.trials = 100,
  epochs = NULL,
  CV.size = 0.25,
  ts.test = NULL,
  folds = 5,
  threshold = NULL,
  obj.fn = expression(sum((predicted - actual)^2)),
  objective = "min",
  extreme = FALSE,
  feature.importance = TRUE,
  status = TRUE,
  ncores = NULL
)
```

## 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), set to (type = "CLASS"), else for continuous (DV.train) set to (type = NULL).

<code>representative.sample</code>	logical; FALSE (default) Reduces observations of <code>IVs.train</code> to a set of representative observations per regressor.
<code>depth</code>	options: (integer, NULL, "max"); Specifies the order parameter in the <a href="#">NNS.reg</a> routine, assigning a number of splits in the regressors. ( <code>depth = "max"</code> ) (default) will be significantly faster, but increase the variance of results, which is suggested for mixed continuous and discrete (unordered, ordered) data.
<code>n.best</code>	integer; NULL (default) Sets the number of nearest regression points to use in weighting for multivariate regression at $\sqrt{\text{\# of regressors}}$ . Analogous to $k$ in a $k$ Nearest Neighbors algorithm. If NULL, determines the optimal clusters via the <a href="#">NNS.stack</a> procedure.
<code>learner.trials</code>	integer; NULL (default) Sets the number of trials to obtain an accuracy threshold level. ( <code>learner.trials = 100</code> ) is the default setting.
<code>epochs</code>	integer; $2 \times \text{length}(\text{DV.train})$ (default) Total number of feature combinations to run.
<code>CV.size</code>	numeric [0, 1]; ( <code>CV.size = .25</code> ) (default) Sets the cross-validation size. Defaults to 0.25 for a 25 percent random sampling of the training set.
<code>ts.test</code>	integer; NULL (default) Sets the length of the test set for time-series data; typically $2 \times h$ parameter value from <a href="#">NNS.ARMA</a> or double known periods to forecast.
<code>folds</code>	integer; 5 (default) Sets the number of folds in the <a href="#">NNS.stack</a> procedure for optimal <code>n.best</code> parameter.
<code>threshold</code>	numeric; NULL (default) Sets the <code>obj.fn</code> threshold to keep feature combinations.
<code>obj.fn</code>	expression; <code>expression( sum((predicted - actual)^2) )</code> (default) Sum of squared errors is the default objective function. Any <code>expression()</code> using the specific terms predicted and actual can be used. Automatically selects an accuracy measure when ( <code>type = "CLASS"</code> ).
<code>objective</code>	options: ("min", "max") "max" (default) Select whether to minimize or maximize the objective function <code>obj.fn</code> .
<code>extreme</code>	logical; FALSE (default) Uses the maximum (minimum) threshold obtained from the <code>learner.trials</code> , rather than the upper (lower) quintile level for maximization (minimization) objective.
<code>feature.importance</code>	logical; TRUE (default) Plots the frequency of features used in the final estimate.
<code>status</code>	logical; TRUE (default) Prints status update message in console.
<code>ncores</code>	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.

**Value**

Returns a vector of fitted values for the dependent variable test set `$results`, and the final feature loadings `$feature.weights`.

**Note**

Like a logistic regression, the (`type = "CLASS"`) setting is not necessary for target variable of two classes e.g. [0, 1].

**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:
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")

## 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, factor.2.dummy = FALSE, tau = 0, plot = FALSE)
```

**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.

**Value**

Returns the directional causation ( $x \rightarrow y$ ) or ( $y \rightarrow x$ ) and net quantity of association. For causal matrix, directional causation is returned as ([column variable]  $\rightarrow$  [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"  
<https://www.amazon.com/dp/1490523995>

**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	integer; (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"  
<https://www.amazon.com/dp/1490523995>

Viole, F. (2017) "Continuous CDFs and ANOVA with NNS" <https://ssrn.com/abstract=3007373>

**Examples**

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

## Not run:
## 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 = c(0,0))

## 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 dimensions to x.
order	integer; Controls the level of quadrant partitioning. Defaults 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.

**Note**

p-values and confidence intervals can be obtained from sampling random permutations of `y_p` and running `NNS.dep(x, y_p)` to compare against a null hypothesis of 0 correlation or independence between `x`, `y`.

See [NNS.dep](#) for examples.

**Author(s)**

Fred Viole, OVVO Financial Systems

**References**

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments"  
<https://www.amazon.com/dp/1490523995>

**Examples**

```
## Not run:
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)
## 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,
  order = 3,
  degree = NULL,
  asym = FALSE,
  print.map = FALSE,
  ncores = NULL
)
```

**Arguments**

x	a numeric vector, matrix or data frame.
y	NULL (default) or a numeric vector with compatible dimensions to x.
order	integer; Controls the level of quadrant partitioning. Defaults to (order = 3). Errors can generally be rectified by setting (order = 1).
degree	integer; Defaults to NULL to allow number of observations to be "degree" determinant.
asym	logical; FALSE (default) Allows for asymmetrical dependencies.
print.map	logical; FALSE (default) Plots quadrant means.
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.

**Value**

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

**Note**

p-values and confidence intervals can be obtained from sampling random permutations of y\_p and running `NNS.dep(x, y_p)` to compare against a null hypothesis of 0 correlation or independence between x, y.

**Author(s)**

Fred Viole, OVVO Financial Systems

**References**

Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments"  
<https://www.amazon.com/dp/1490523995>

**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)

## p-values for [NNS.dep]
x <- seq(-5, 5, .1); y <- x^2 + rnorm(length(x))

nns_cor_dep <- NNS.dep(x, y, print.map = TRUE)
nns_cor_dep
```

```

## Create permutations of y
y_p <- replicate(1000, sample.int(length(y)))

## Generate new correlation and dependence measures on each new permutation of y
nns.mc <- apply(y_p, 2, function(g) NNS.dep(x, y[g]))

## Store results
cors <- unlist(lapply(nns.mc, "[", 1))
deps <- unlist(lapply(nns.mc, "[", 2))

## View results
hist(cors)
hist(deps)

## Left tailed correlation p-value
cor_p_value <- LPM(0, nns_cor_dep$Correlation, cors)
cor_p_value

## Right tailed correlation p-value
cor_p_value <- UPM(0, nns_cor_dep$Correlation, cors)
cor_p_value

## Confidence Intervals
## For 95th percentile VaR (both-tails) see [LPM.VaR] and [UPM.VaR]
## Lower CI
LPM.VaR(.025, 0, cors)
## Upper CI
UPM.VaR(.025, 0, cors)

## Left tailed dependence p-value
dep_p_value <- LPM(0, nns_cor_dep$Dependence, deps)
dep_p_value

## Right tailed dependence p-value
dep_p_value <- UPM(0, nns_cor_dep$Dependence, deps)
dep_p_value

## End(Not run)

```

---

NNS.dep.base

*NNS Dependence Base*


---

## Description

Internal function for NNS dependence [NNS.dep](#) parallel instances.

## Usage

```

NNS.dep.base(
  x,
  y = NULL,
  order = NULL,
  degree = NULL,
  type = NULL,

```



```

    print.map = FALSE,
    asym = FALSE
  )

```

### Arguments

x	from <a href="#">NNS.part</a>
y	from <a href="#">NNS.part</a>
order	from <a href="#">NNS.part</a>
degree	from <a href="#">NNS.part</a>
type	from <a href="#">NNS.part</a>
print.map	from <a href="#">NNS.part</a>
asym	for asymmetrical dependencies

### Value

Returns NNS dependence.

---

NNS.dep.hd

*NNS Co-Partial Moments Higher Dimension Dependence*


---

### Description

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

### Usage

```
NNS.dep.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 <a href="#">plot3d</a> .
independence.overlay	logical; FALSE (default) Creates and overlays independent <a href="#">Co.LPM</a> and <a href="#">Co.UPM</a> 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

- \$actual.observations Number of [Co.LPM](#) and [Co.UPM](#) observations.
- \$independent.null Expected number of [Co.LPM](#) and [Co.UPM](#) observations under the null hypothesis of independence.
- \$Dependence Multivariate nonlinear dependence coefficient [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" <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.dep.hd(A, plot = TRUE, independence.overlay = TRUE)
```

---

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. Defaults 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" <https://www.amazon.com/dp/1490523995>

## Examples

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

---

NNS.distance	<i>NNS Distance</i>
--------------	---------------------

---

**Description**

Internal function for NNS multivariate regression [NNS.reg](#) parallel instances.

**Usage**

```
NNS.distance(rpm, dist.estimate, type, k)
```

**Arguments**

rpm	REGRESSION.POINT.MATRIX from <a href="#">NNS.reg</a>
dist.estimate	Vector to generate distances from.
type	"L1", "L2" or "DTW"
k	n.best from <a href="#">NNS.reg</a>

**Value**

Returns sum of weighted distances.

---

NNS.FSD	<i>NNS FSD Test</i>
---------	---------------------

---

**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.meboot

*NNS meboot***Description**

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

**Usage**

```
NNS.meboot(
  x,
  reps = 999,
  setSpearman = NULL,
  drift = TRUE,
  trim = 0.1,
  xmin = NULL,
  xmax = NULL,
  reachbnd = TRUE,
  expand.sd = TRUE,
  force.clt = TRUE,
  scl.adjustment = FALSE,
  sym = FALSE,
  elaps = FALSE,
  colsubj,
  coldata,
  coltimes,
  ...
)
```

**Arguments**

<code>x</code>	vector of data, ts object or pdata.frame object.
<code>reps</code>	numeric; number of replicates to generate.
<code>setSpearman</code>	numeric [0,1]; The default setting <code>setSpearman = NULL</code> assumes that the user does not want to generate replicates that are perfectly dependent on original time series, <code>setSpearman=1</code> recovers the original <code>meboot(...)</code> settings. <code>setSpearman &lt; 1</code> admits less perfect (more realistic for some purposes) dependence.
<code>drift</code>	logical; TRUE default preserves the drift of the original series.
<code>trim</code>	numeric [0,1]; The mean trimming proportion, defaults to <code>trim=0.1</code> .
<code>xmin</code>	numeric; the lower limit for the left tail.
<code>xmax</code>	numeric; the upper limit for the right tail.
<code>reachbnd</code>	logical; If TRUE potentially reached bounds ( <code>xmin</code> = smallest value - trimmed mean and <code>xmax</code> = largest value + trimmed mean) are given when the random draw happens to be equal to 0 and 1, respectively.
<code>expand.sd</code>	logical; If TRUE the standard deviation in the ensemble is expanded. See <code>expand.sd</code> in <code>meboot::meboot</code> .
<code>force.clt</code>	logical; If TRUE the ensemble is forced to satisfy the central limit theorem. See <code>force.clt</code> in <code>meboot::meboot</code> .

<code>scl.adjustment</code>	logical; If TRUE scale adjustment is performed to ensure that the population variance of the transformed series equals the variance of the data.
<code>sym</code>	logical; If TRUE an adjustment is performed to ensure that the ME density is symmetric.
<code>elaps</code>	logical; If TRUE elapsed time during computations is displayed.
<code>colsubj</code>	numeric; the column in <code>x</code> that contains the individual index. It is ignored if the input data <code>x</code> is not a <code>pdata.frame</code> object.
<code>coldata</code>	numeric; the column in <code>x</code> that contains the data of the variable to create the ensemble. It is ignored if the input data <code>x</code> is not a <code>pdata.frame</code> object.
<code>coltimes</code>	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 <code>x</code> is not a <code>pdata.frame</code> object.
<code>...</code>	possible argument <code>fiv</code> to be passed to <code>expand.sd</code> .

### Value

- `x` original data provided as input.
- replicates maximum entropy bootstrap replicates.
- ensemble average observation over all 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.

### References

- Vinod, H.D. (2013), Maximum Entropy Bootstrap Algorithm Enhancements. <http://ssrn.com/abstract=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, setSpearman = 0, xmin = 0)

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

# Plot all replicates
```

```

matplot(boots$replicates, type = 'l')

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

## End(Not run)

```

NNS.norm

*NNS Normalization***Description**

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

**Usage**

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

**Arguments**

A	a numeric matrix or data frame.
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 <a href="#">legend</a> .

**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"  
<https://www.amazon.com/dp/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,
  obs.req = 10,
  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; (10 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: ("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.

**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"  
<https://www.amazon.com/dp/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 [dy.dx](#) on [NNS.CDF](#).

### 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 number of 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"  
<https://www.amazon.com/dp/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 = 0.95,
  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 = "off",
  dist = "L2",
```

```

    ncores = NULL,
    multivariate.call = FALSE
  )

```

## Arguments

<code>x</code>	a vector, matrix or data frame of variables of numeric or factor data types.
<code>y</code>	a numeric or factor vector with compatible dimensions to <code>x</code> .
<code>factor.2.dummy</code>	logical; TRUE (default) Automatically augments variable matrix with numerical dummy variables based on the levels of factors.
<code>order</code>	integer; Controls the number of partial moment quadrant means. Users are encouraged to try different ( <code>order = ...</code> ) integer settings with ( <code>noise.reduction = "off"</code> ). ( <code>order = "max"</code> ) will force a limit condition perfect fit.
<code>stn</code>	numeric [0, 1]; Signal to noise parameter, sets the threshold of ( <code>NNS.dep</code> ) which reduces (" <code>order</code> ") when ( <code>order = NULL</code> ). Defaults to 0.95 to ensure high dependence for higher (" <code>order</code> ") and endpoint determination.
<code>dim.red.method</code>	options: (" <code>cor</code> ", " <code>NNS.dep</code> ", " <code>NNS.caus</code> ", " <code>all</code> ", <code>NULL</code> ) method for determining synthetic $X^*$ coefficients. Selection of a method automatically engages the dimension reduction regression. The default is <code>NULL</code> for full multivariate regression. ( <code>dim.red.method = "NNS.dep"</code> ) uses <a href="#">NNS.dep</a> for nonlinear dependence weights, while ( <code>dim.red.method = "NNS.caus"</code> ) uses <a href="#">NNS.caus</a> for causal weights. ( <code>dim.red.method = "cor"</code> ) uses standard linear correlation for weights. ( <code>dim.red.method = "all"</code> ) averages all methods for further feature engineering.
<code>tau</code>	options(" <code>ts</code> ", <code>NULL</code> ); <code>NULL</code> (default) To be used in conjunction with ( <code>dim.red.method = "NNS.caus"</code> ) or ( <code>dim.red.method = "all"</code> ). If the regression is using time-series data, set ( <code>tau = "ts"</code> ) for more accurate causal analysis.
<code>type</code>	<code>NULL</code> (default). To perform a classification, set to ( <code>type = "CLASS"</code> ). Like a logistic regression, it is not necessary for target variable of two classes e.g. [0, 1].
<code>point.est</code>	a numeric or factor vector with compatible dimensions to <code>x</code> . Returns the fitted value $\hat{y}$ for any value of <code>x</code> .
<code>location</code>	Sets the legend location within the plot, per the <code>x</code> and <code>y</code> co-ordinates used in base graphics <a href="#">legend</a> .
<code>return.values</code>	logical; TRUE (default), set to FALSE in order to only display a regression plot and call values as needed.
<code>plot</code>	logical; TRUE (default) To plot regression.
<code>plot.regions</code>	logical; FALSE (default). Generates 3d regions associated with each regression point for multivariate regressions. Note, adds significant time to routine.
<code>residual.plot</code>	logical; TRUE (default) To plot $\hat{y}$ and $Y$ .
<code>std.errors</code>	logical; FALSE (default) To provide standard errors of each linear segment in the " <code>Fitted.xy</code> " output.
<code>confidence.interval</code>	numeric [0, 1]; <code>NULL</code> (default) Plots the associated confidence interval with the estimate and reports the standard error for each individual segment.
<code>threshold</code>	numeric [0, 1]; ( <code>threshold = 0</code> ) (default) Sets the threshold for dimension reduction of independent variables when ( <code>dim.red.method</code> ) is not <code>NULL</code> .

n.best	integer; NULL (default) Sets the number of nearest regression points to use in weighting for multivariate regression at $\sqrt{\text{# of regressors}}$ . (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 <a href="#">NNS.stack</a> .
noise.reduction	the method of determining regression points options: ("mean", "median", "mode", "off"); In low signal:noise situations, (noise.reduction = "mean") uses means for <a href="#">NNS.dep</a> restricted partitions, (noise.reduction = "median") uses medians instead of means for <a href="#">NNS.dep</a> restricted partitions, while (noise.reduction = "mode") uses modes instead of means for <a href="#">NNS.dep</a> restricted partitions. (noise.reduction = "off") uses an overall central tendency measure for partitions.
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.
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.
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.est" for the predicted value generated;
- "regression.points" provides the points used in the regression equation for the given order of partitions;
- "Fitted.xy" returns a [data.table](#) 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.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.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.

**Author(s)**

Fred Viole, OVVO Financial Systems

**References**

- Viole, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" <https://www.amazon.com/dp/1490523995>
- Vinod, H. and Viole, F. (2017) "Nonparametric Regression Using Clusters" <https://link.springer.com/article/10.1007/s10614-017-9713-5>
- Vinod, H. and Viole, F. (2018) "Clustering and Curve Fitting by Line Segments" <https://www.preprints.org/manuscript/201801.0090/v1>

**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 partitioning (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")

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

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

# 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.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	<i>NNS Seasonality Test</i>
----------	-----------------------------

---

**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; code 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"  
<https://www.amazon.com/dp/1490523995>

**Examples**

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

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

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

---

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,
  type = NULL,
  obj.fn = expression(sum((predicted - actual)^2)),
  objective = "min",
  dist = "L2",
  CV.size = NULL,
  ts.test = NULL,
  folds = 5,
  order = NULL,
  norm = NULL,
  method = c(1, 2),
  stack = TRUE,
  dim.red.method = "cor",
  status = TRUE,
  ncores = NULL
)
```

**Arguments**

<code>IVs.train</code>	a vector, matrix or data frame of variables of numeric or factor data types.
<code>DV.train</code>	a numeric or factor vector with compatible dimensions to ( <code>IVs.train</code> ).
<code>IVs.test</code>	a vector, matrix or data frame of variables of numeric or factor data types with compatible dimensions to ( <code>IVs.train</code> ). If NULL, will use ( <code>IVs.train</code> ) as default.
<code>type</code>	NULL (default). To perform a classification of discrete integer classes from factor target variable ( <code>DV.train</code> ), set to ( <code>type = "CLASS"</code> ), else for continuous ( <code>DV.train</code> ) set to ( <code>type = NULL</code> ). Like a logistic regression, this setting is not necessary for target variable of two classes e.g. [0, 1].
<code>obj.fn</code>	expression; <code>expression(sum((predicted - actual)^2))</code> (default) Sum of squared errors is the default objective function. Any <code>expression()</code> using the specific terms predicted and actual can be used.
<code>objective</code>	options: ("min", "max") "min" (default) Select whether to minimize or maximize the objective function <code>obj.fn</code> .
<code>dist</code>	options: ("L1", "L2", "DTW", "FACTOR") the method of distance calculation; Selects the distance calculation used. <code>dist = "L2"</code> (default) selects the Euclidean distance and ( <code>dist = "L1"</code> ) selects the Manhattan distance; ( <code>dist = "DTW"</code> ) selects the dynamic time warping distance; ( <code>dist = "FACTOR"</code> ) uses a frequency.
<code>CV.size</code>	numeric [0, 1]; NULL (default) Sets the cross-validation size if ( <code>IVs.test = NULL</code> ). Defaults to 0.25 for a 25 percent random sampling of the training set under ( <code>CV.size = NULL</code> ).
<code>ts.test</code>	integer; NULL (default) Sets the length of the test set for time-series data; typically $2 \times h$ parameter value from <a href="#">NNS.ARMA</a> or double known periods to forecast.
<code>folds</code>	integer; <code>folds = 5</code> (default) Select the number of cross-validation folds.
<code>order</code>	options: (integer, "max", NULL); NULL (default) Sets the order for <a href="#">NNS.reg</a> , where ( <code>order = "max"</code> ) is the k-nearest neighbors equivalent, which is suggested for mixed continuous and discrete (unordered, ordered) data.
<code>norm</code>	options: ("std", "NNS", NULL); NULL (default) 3 settings offered: NULL, "std", and "NNS". Selects the norm parameter in <a href="#">NNS.reg</a> .
<code>method</code>	numeric options: (1, 2); Select the NNS method to include in stack. ( <code>method = 1</code> ) selects <a href="#">NNS.reg</a> ; ( <code>method = 2</code> ) selects <a href="#">NNS.reg</a> dimension reduction regression. Defaults to <code>method = c(1, 2)</code> , which will reduce the dimension first, then find the optimal <code>n.best</code> .
<code>stack</code>	logical; TRUE (default) Uses dimension reduction output in <code>n.best</code> optimization, otherwise performs both analyses independently.
<code>dim.red.method</code>	options: ("cor", "NNS.dep", "NNS.caus", "all") method for determining synthetic $X^*$ coefficients. ( <code>dim.red.method = "cor"</code> ) (default) uses standard linear correlation for weights. ( <code>dim.red.method = "NNS.dep"</code> ) uses <a href="#">NNS.dep</a> for nonlinear dependence weights, while ( <code>dim.red.method = "NNS.caus"</code> ) uses <a href="#">NNS.caus</a> for causal weights. ( <code>dim.red.method = "all"</code> ) averages all methods for further feature engineering.
<code>status</code>	logical; TRUE (default) Prints status update message in console.
<code>ncores</code>	integer; value specifying the number of cores to be used in the parallelized subroutine <a href="#">NNS.reg</a> . 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.
- "reg" returns [NNS.reg](#) output.
- "dim.red" returns [NNS.reg](#) dimension reduction regression output.
- "stack" returns the output of the stacked model.

**Note**

- Like a logistic regression, the (type = "CLASS") setting is not necessary for target variable of two classes e.g. [0, 1].
- 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 Violen, OVVO Financial Systems

**References**

Violen, 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], type = "CLASS")

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

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

---

NNS.term.matrix	<i>NNS Term Matrix</i>
-----------------	------------------------

---

## 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 possibility 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" <https://www.amazon.com/dp/1490523995>

## Examples

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

### Concatenate Text with space separator, 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], NYT[, 4], sep = " "),
                      as.numeric(as.character(NYTimes[, 5]))))
NNS.term.matrix(NYT)
## End(Not run)
```

---

NNS.TSD	<i>NNS TSD Test</i>
---------	---------------------

---

**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	<i>NNS TSD Test uni-directional</i>
-------------	-------------------------------------

---

**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)
```

---

NNS.VAR	<i>NNS VAR</i>
---------	----------------

---

**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",
  obj.fn = expression(sum((predicted - actual)^2)),
  objective = "min",
  status = TRUE,
  ncores = NULL
)
```

**Arguments**

variables	a numeric matrix or data.frame of contemporaneous time-series to forecast.
h	integer; 1 (default) Number of periods to forecast.
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 <a href="#">NNS.stack</a> . (dim.red.method = "cor") (default) uses standard linear correlation for dimension reduction in the lagged variable matrix. (dim.red.method = "NNS.dep") uses <a href="#">NNS.dep</a> for nonlinear dependence weights, while (dim.red.method = "NNS.caus") uses <a href="#">NNS.caus</a> for causal weights. (dim.red.method = "all") averages all methods for further feature engineering.
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.
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 <a href="#">NNS.ARMA.optim</a> . If NULL (default), the number of cores to be used is equal to the number of cores of the machine - 1.

**Value**

Returns the following matrices of forecasted variables:

- "interpolated\_and\_extrapolated" Returns a [data.table](#) 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**

- dim.red.method = "cor" is significantly faster than the other methods, but comes at the expense of ignoring possible nonlinear relationships between lagged variables.
- 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" <https://www.amazon.com/dp/1490523995>
- Viole, F. (2019) "Multi-variate Time-Series Forecasting: Nonparametric Vector Autoregression Using NNS" <https://ssrn.com/abstract=3489550>
- Viole, F. (2020) "NOWCASTING with NNS" <https://ssrn.com/abstract=3586658>
- Viole, F. (2019) "Forecasting Using NNS" <https://ssrn.com/abstract=3382300>
- Vinod, H. and Viole, F. (2017) "Nonparametric Regression Using Clusters" <https://link.springer.com/article/10.1007/s10614-017-9713-5>
- Vinod, H. and Viole, F. (2018) "Clustering and Curve Fitting by Line Segments" <https://www.preprints.org/manuscript/201801.0090/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)

#####
### 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	<i>Partial Moment Matrix</i>
-----------	------------------------------

---

**Description**

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

**Usage**

```
PM.matrix(LPM.degree, UPM.degree, target = "mean", 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") (default) 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 (CUPM, DUPM, DLPM, CLPM), and overall covariance matrix. 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 Violen, OVVO Financial Systems

**References**

Violen, F. and Nawrocki, D. (2013) "Nonlinear Nonparametric Statistics: Using Partial Moments" <https://www.amazon.com/dp/1490523995>  
 Violen, F. (2017) "Bayes' Theorem From Partial Moments" <https://ssrn.com/abstract=3457377>

**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)

## Use of vectorized numeric targets (target_x, target_y, target_z)
```

```

PM.matrix(LPM.degree = 1, UPM.degree = 1, target = c(0, 0.15, .25), 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$cov.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"  
<https://www.amazon.com/dp/1490523995>

**Examples**

```

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

```

---

UPM.ratio	<i>Upper Partial Moment RATIO</i>
-----------	-----------------------------------

---

### 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"  
<https://www.amazon.com/dp/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 value at risk (VaR) quantile based on the Upper Partial Moment ratio

**Usage**

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

**Arguments**

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

**Value**

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

**Examples**

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

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

# Index

Co.LPM, [2](#), [25](#)  
Co.UPM, [3](#), [25](#)  
complete.cases, [43](#)

D.LPM, [4](#)  
D.UPM, [5](#)  
data.frame, [31](#)  
data.table, [14](#), [33](#), [36](#), [47](#)  
dy.d\_, [7](#)  
dy.dx, [6](#), [33](#)

legend, [31](#), [35](#)  
LPM, [9](#)  
LPM.ratio, [9](#), [20](#)  
LPM.VaR, [10](#)

meboot, [14](#)

na.omit, [43](#)  
NNS.ANOVA, [11](#)  
NNS.ARMA, [13](#), [15](#), [16](#), [18](#), [42](#), [46](#), [47](#)  
NNS.ARMA.optim, [15](#), [39](#), [47](#)  
NNS.boost, [17](#)  
NNS.caus, [19](#), [35](#), [42](#), [47](#)  
NNS.CDF, [20](#), [33](#)  
NNS.cor, [21](#), [32](#)  
NNS.dep, [22](#), [22](#), [24](#), [32](#), [35](#), [36](#), [42](#), [47](#)  
NNS.dep.base, [24](#)  
NNS.dep.hd, [25](#)  
NNS.diff, [26](#)  
NNS.distance, [27](#)  
NNS.FSD, [27](#)  
NNS.FSD.uni, [28](#)  
NNS.meboot, [29](#)  
NNS.norm, [31](#)  
NNS.part, [25](#), [32](#)  
NNS.PDF, [33](#)  
NNS.reg, [6–8](#), [17](#), [18](#), [27](#), [32](#), [34](#), [41–44](#), [46](#), [47](#)  
NNS.SD.efficient.set, [38](#)  
NNS.seas, [13](#), [39](#)  
NNS.SSD, [40](#)  
NNS.SSD.uni, [40](#)  
NNS.stack, [7](#), [18](#), [36](#), [41](#), [47](#)  
NNS.term.matrix, [44](#)

NNS.TSD, [45](#)  
NNS.TSD.uni, [45](#)  
NNS.VAR, [46](#)

plot3d, [25](#)  
PM.matrix, [49](#)

UPM, [50](#)  
UPM.ratio, [51](#)  
UPM.VaR, [52](#)