

# Package ‘NNS’

July 16, 2017

**Type** Package

**Title** Nonlinear Nonparametric Statistics

**Version** 0.3.5

**Date** 2017-07-16

**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 modelling, 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** 6.0.1

**Depends** R ( $\geq 3.3.0$ ), data.table, rgl, stringr

**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 . . . . .	8
LPM.VaR . . . . .	9
NNS.ANOVA . . . . .	10
NNS.ARMA . . . . .	11
NNS.caus . . . . .	13
NNS.cor . . . . .	14
NNS.cor.hd . . . . .	15
NNS.dep . . . . .	16

NNS.diff . . . . .	17
NNS.FSD . . . . .	18
NNS.FSD.uni . . . . .	18
NNS.norm . . . . .	19
NNS.part . . . . .	20
NNS.reg . . . . .	21
NNS.SD.Efficient.Set . . . . .	24
NNS.seas . . . . .	25
NNS.SSD . . . . .	26
NNS.SSD.uni . . . . .	27
NNS.stack . . . . .	27
NNS.term.matrix . . . . .	29
NNS.TSD . . . . .	30
NNS.TSD.uni . . . . .	31
PM.matrix . . . . .	31
UPM . . . . .	32
UPM.VaR . . . . .	33

## Index 35

---

Co.LPM	<i>Co-Lower Partial Moment (Lower Left Quadrant 4)</i>
--------	--

---

### 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, target.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	<i>Co-Upper Partial Moment (Upper Right Quadrant 1)</i>
--------	---

---

## 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, target.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, target.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, target.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 <a href="#">NNS.reg</a> . Default setting is (noise.reduction="mean").
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.

**Author(s)**

Fred Viole, OVVO Financial Systems

## References

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

Vinod, H. and Viola, 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)
```

---

dy.d_	<i>Partial Derivative dy/d[wrt]</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", 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 dimensions 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.
order	integer; <a href="#">NNS.reg</a> "order", defaults to NULL.
stn	numeric [0,1]; Signal to noise parameter, sets the threshold of <a href="#">NNS.dep</a> 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 <a href="#">NNS.reg</a> . NULL (default) Uses <code>ceiling(sqrt(ncol(x)))</code> .
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.
noise.reduction	the method of determining 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 <a href="#">NNS.reg</a> . 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).

**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]
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")
```

---

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.VaR	<i>LPM VaR</i>
---------	----------------

---

**Description**

Generates a VaR based on the Lower Partial Moment ratio

**Usage**

```
LPM.VaR(percentile, degree, x, extend = NULL)
```

**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.
extend	options: ("yes", NULL); NULL (default) Sets the "extendInt" argument from <a href="#">uniroot</a> .

**Value**

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

**Note**

If endpoint error is generated, set (extend="yes").

**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	<i>NNS ANOVA</i>
-----------	------------------

---

## 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, pairwise = FALSE,
plot = TRUE, binary = TRUE, extend = NULL)
```

## 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).
pairwise	logical; FALSE (default) 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.
extend	options: ("yes", NULL); NULL (default) Sets the "extendInt" argument from <a href="#">uniroot</a> .

## 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

### Note

If endpoint error is generated, set (extend="yes").

### 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 = NULL, negative.values = FALSE, method = "nonlin",
  dynamic = FALSE, plot = TRUE, seasonal.plot = TRUE, intervals = FALSE)
```

**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>best.periods</code>	integer; to be 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> ).
<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>intervals</code>	logical; FALSE (default) Plots the surrounding forecasts around the final estimate when ( <code>intervals=TRUE</code> ) and ( <code>seasonal.factor=FALSE</code> ). There are no other forecasts to plot when a single <code>seasonal.factor</code> is selected.

**Value**

Returns a vector of forecasts of length (h).

**Note**

(`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

Violo, 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)

## Nonlinear NNS.ARMA using AirPassengers monthly data and 12, 24, and 36 period lags
## Not run:
NNS.ARMA(AirPassengers,h=45,training.set=100,seasonal.factor=c(12,24,36),method='nonlin')
## End(Not run)

## Nonlinear NNS.ARMA using AirPassengers monthly data and 2 best periods lag
## Not run:
NNS.ARMA(AirPassengers,h=45,training.set=100,seasonal.factor=FALSE,best.periods=2,method='nonlin')
## End(Not run)
```

---

NNS.caus	<i>NNS Causation</i>
----------	----------------------

---

## Description

Returns the causality from observational data between two variables

## Usage

```
NNS.caus(x, y, tau, plot = FALSE)
```

## Arguments

x	a numeric vector, matrix or data frame.
y	NULL (default) or a numeric vector with compatible dimensions to x.
tau	integer; Number of lagged observations to consider.
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, gross quantity of association is returned as ( $x[\text{column}] \rightarrow y[\text{row}]$ ).

## Author(s)

Fred Violo, OVVO Financial Systems

## References

Viola, 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,1)

## 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 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.

## Author(s)

Fred Viola, OVVO Financial Systems

## References

Viola, 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 <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**

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 dimensions 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)
```



**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**

<code>f</code>	an expression or call or a formula with no lhs.
<code>point</code>	numeric; Point to be evaluated for derivative of a given function <code>f</code> .
<code>h</code>	numeric [0, ...]; Initial step for secant projection. Defaults to ( $h=0.1$ ).
<code>tol</code>	numeric; Sets the tolerance for the stopping condition of the inferred $h$ . Defaults to ( $tol=1e-10$ ).
<code>print.trace</code>	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>.

**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**

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

**Value**

Returns (1) if " $X \text{ FSD } Y$ ", 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>.

**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**

<code>A</code>	a numeric matrix or data frame.
<code>chart.type</code>	options: ("l", "b"); NULL (default). Set ( <code>chart.type="l"</code> ) for line, ( <code>chart.type="b"</code> ) for boxplot.
<code>linear</code>	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	<i>NNS Partition Map</i>
----------	--------------------------

---

## 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 = 4,
min.obs.stop = FALSE, noise.reduction = "mean")
```

## 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.
max.obs	integer; (4 default) Desired 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=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.
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.reg

*NNS Regression*


---

**Description**

Generates a nonlinear regression based on partial moment quadrant means.

**Usage**

```
NNS.reg(x, y, order = NULL, stn = 0.99, dim.red.method = NULL,
  type = NULL, point.est = NULL, location = "top", return.values = TRUE,
  plot = TRUE, plot.regions = FALSE, residual.plot = TRUE,
  threshold = 0, n.best = NULL, noise.reduction = "mean", norm = NULL,
  dist = "L2", 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>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 0.99 to ensure high dependence for higher (" <code>order</code> ") and endpoint determination. ( <code>noise.reduction="off"</code> ) sets ( <code>stn=0</code> ) to allow for maximum fit.
<code>dim.red.method</code>	options: (" <code>cor</code> ", " <code>NNS.cor</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.cor"</code> ) uses <a href="#">NNS.cor</a> for nonlinear correlation 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>type</code>	<code>NULL</code> (default). To perform logistic regression, set to ( <code>type = "LOGIT"</code> ). To perform a classification, set to ( <code>type = "CLASS"</code> ).
<code>point.est</code>	a numeric or factor vector with compatible dimensions to <code>x</code> . Returns the fitted value <code>y.hat</code> 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; <code>TRUE</code> (default), set to <code>FALSE</code> in order to only display a regression plot and call values as needed.
<code>plot</code>	logical; <code>TRUE</code> (default) To plot regression.
<code>plot.regions</code>	logical; <code>FALSE</code> (default). Generates 3d regions associated with each regression point for multivariate regressions. Note, adds significant time to routine.
<code>residual.plot</code>	logical; <code>TRUE</code> (default) To plot <code>y.hat</code> and <code>Y</code> .
<code>threshold</code>	numeric [0,1]; <code>threshold=0</code> (default) Sets the correlation threshold for independent variables.
<code>n.best</code>	integer; <code>NULL</code> (default) Sets the number of nearest regression points to use in weighting for multivariate regression at $2*(\# \text{ of regressors})$ . ( <code>n.best="all"</code> ) will select and weight all generated regression points. Analogous to <code>k</code> in <code>k</code> Nearest Neighbors algorithm and different values are tested using cross-validation in <a href="#">NNS.stack</a> .
<code>noise.reduction</code>	the method of determining regression points options: (" <code>mean</code> ", " <code>median</code> ", " <code>mode</code> ", " <code>off</code> "); In low signal:noise situations, ( <code>noise.reduction="mean"</code> ) uses means for <a href="#">NNS.dep</a> restricted partitions, ( <code>noise.reduction="median"</code> ) uses medians instead of means for <a href="#">NNS.dep</a> restricted partitions, while ( <code>noise.reduction="mode"</code> ) uses modes instead of means for <a href="#">NNS.dep</a> restricted partitions. ( <code>noise.reduction="off"</code> ) allows for maximum possible fit with a specific order.
<code>norm</code>	<code>NULL</code> (default) the method of normalization options: (" <code>NNS</code> ", " <code>std</code> "); Normalizes <code>x</code> between 0 and 1 for multivariate regression when set to ( <code>norm="std"</code> ), or normalizes <code>x</code> according to <a href="#">NNS.norm</a> when set to ( <code>norm="NNS"</code> ).

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

`multivariate.call` Internal parameter for multivariate regressions.

## Value

UNIVARIATE REGRESSION RETURNS THE FOLLOWING VALUES:

- "R2" provides the goodness of fit;
- "MSE" returns the MSE 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  $> 0$ .
- "`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`, 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 ( $\geq v.0.3.4$ ) has repurposed parameter (`type="CLASS"`). (`type="CLASS"`) is now restricted to signifying a classification analysis for `NNS.reg` while (`dim.red=TRUE`) 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
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")

## Logistic Regression (Univariate only)
NNS.reg(x,y,type="LOGIT")

## 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")

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

## To call partial derivative (univariate regression only):
x<-rnorm(100); y<-rnorm(100)
NNS.reg(x,y)$derivative
```



**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>.

**Examples**

```
set.seed(123)
x<-rnorm(100); y<-rnorm(100); z<-rnorm(100)
x<-data.frame(x,y,z)
NNS.SD.Efficient.Set(x,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	<i>NNS SSD Test</i>
---------	---------------------

---

**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

**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,
  weight = "MSE", 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 dimensions 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).
weight	options: ("MSE","Features") method for selecting model output weight; Set (weight="MSE") for optimum parameters and weighting based on each base model's "MSE". (weight="Features") uses a weighting based on the number of features present, whereby logistic <a href="#">NNS.reg</a> receives higher relative weights for more regressors. Defaults to "MSE".
norm	options: ("std","NNS", NULL); NULL (default) 3 settings offered: NULL, "std", and "NNS". Selects the norm parameter in <a href="#">NNS.reg</a> .
method	numeric options: (1,2); Select the NNS method to include in stack. (method=1) selects <a href="#">NNS.reg</a> ; (method=2) selects <a href="#">NNS.reg</a> 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 <a href="#">NNS.cor</a> for non-linear correlation 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.
seed	numeric; 123 (default) Sets seed for CV sampling.

## 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. "MSE.reg" returns the MSE for the [NNS.reg](#) multivariate regression.
- "NNS.dim.red.threshold" returns the optimum "threshold" from the [NNS.reg](#) dimension reduction regression.
- "MSE.dim.red" returns the MSE 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 transformed 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)
```

---

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 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" <http://amzn.com/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],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

**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(Co.PM, degree, target, variable)
```

**Arguments**

Co.PM	options: ("Co.LPM", "Co.UPM", "D.LPM", "D.UPM"); partial moment quadrant to generate matrix of.
degree	integer; Degree for variable. (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.

**Value**

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

**Note**

For "D.LPM" and "D.UPM" matrices, upper triangle of matrix is D.LPM(column,row,...) while lower triangle of matrix is D.LPM(row,column,...).

**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("Co.LPM", degree=0, target="mean", A)

## Calling Multiple Partial Moment Quadrants
PM.matrix(c("Co.LPM","Co.UPM"), degree=0, target="mean", A)
```

---

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.VaR	<i>UPM VaR</i>
---------	----------------

---

**Description**

Generates an upside VaR based on the Upper Partial Moment ratio

**Usage**

```
UPM.VaR(percentile, degree, x, extend = NULL)
```

**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.
extend	options: ("yes", NULL); NULL (default) Sets the "extendInt" argument from <a href="#">uniroot</a> .

**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

- \*Topic **ANOVA**,
  - NNS.ANOVA, [10](#)
- \*Topic **Autoregressive**
  - NNS.ARMA, [11](#)
- \*Topic **CDF**
  - LPM, [8](#)
  - UPM, [32](#)
- \*Topic **VaR**
  - LPM.VaR, [9](#)
  - UPM.VaR, [33](#)
- \*Topic **causation**
  - NNS.caus, [13](#)
- \*Topic **classifier**
  - NNS.reg, [21](#)
  - NNS.stack, [27](#)
- \*Topic **cluster**
  - NNS.part, [20](#)
- \*Topic **correlation**
  - NNS.cor, [14](#)
  - NNS.cor.hd, [15](#)
  - NNS.dep, [16](#)
- \*Topic **covariance**
  - Co.LPM, [2](#)
  - Co.UPM, [3](#)
  - D.LPM, [4](#)
  - D.UPM, [5](#)
  - PM.matrix, [31](#)
- \*Topic **dependence**,
  - NNS.cor.hd, [15](#)
  - NNS.dep, [16](#)
- \*Topic **derivative**
  - dy.d\_, [7](#)
  - dy.dx, [6](#)
- \*Topic **differentiation**,
  - NNS.diff, [17](#)
- \*Topic **document**
  - NNS.term.matrix, [29](#)
- \*Topic **dominance**
  - NNS.FSD, [18](#)
  - NNS.FSD.uni, [18](#)
  - NNS.SD.Efficient.Set, [24](#)
  - NNS.SSD, [26](#)
  - NNS.TSD, [30](#)
- \*Topic **effect**
  - NNS.ANOVA, [10](#)
- \*Topic **matrix**
  - NNS.term.matrix, [29](#)
- \*Topic **mean**,
  - LPM, [8](#)
  - UPM, [32](#)
- \*Topic **model**
  - NNS.ARMA, [11](#)
- \*Topic **moments**,
  - Co.LPM, [2](#)
  - Co.UPM, [3](#)
  - D.LPM, [4](#)
  - D.UPM, [5](#)
  - LPM, [8](#)
  - PM.matrix, [31](#)
  - UPM, [32](#)
- \*Topic **multivariate**
  - dy.d\_, [7](#)
- \*Topic **nonlinear**
  - NNS.cor, [14](#)
  - NNS.reg, [21](#)
- \*Topic **normalization**
  - NNS.norm, [19](#)
- \*Topic **numerical**
  - NNS.diff, [17](#)
- \*Topic **partial**
  - Co.LPM, [2](#)
  - Co.UPM, [3](#)
  - D.LPM, [4](#)
  - D.UPM, [5](#)
  - dy.d\_, [7](#)
  - dy.dx, [6](#)
  - LPM, [8](#)
  - PM.matrix, [31](#)
  - UPM, [32](#)
- \*Topic **partitioning**,
  - NNS.part, [20](#)
- \*Topic **regression**,
  - NNS.reg, [21](#)
- \*Topic **seasonality**
  - NNS.seas, [25](#)
- \*Topic **secant**

- NNS.diff, [17](#)
- \*Topic **size**
  - NNS.ANOVA, [10](#)
- \*Topic **stochastic**
  - NNS.FSD, [18](#)
  - NNS.FSD.uni, [18](#)
  - NNS.SD.Efficient.Set, [24](#)
  - NNS.SSD, [26](#)
  - NNS.TSD, [30](#)
- \*Topic **term**
  - NNS.term.matrix, [29](#)
- \*Topic **upper**
  - UPM, [32](#)
- \*Topic **variance**,
  - LPM, [8](#)
  - UPM, [32](#)
- as.factor, [28](#)
- Co.LPM, [2](#), [15](#)
- Co.UPM, [3](#), [15](#)
- complete.cases, [28](#)
- D.LPM, [4](#)
- D.UPM, [5](#)
- data.frame, [19](#)
- data.matrix, [28](#)
- data.table, [21](#), [23](#)
- dy.d\_, [7](#)
- dy.dx, [6](#)
- expand.grid, [8](#)
- legend, [22](#)
- LPM, [8](#)
- LPM.VaR, [9](#)
- na.omit, [28](#)
- NNS.ANOVA, [10](#)
- NNS.ARMA, [11](#)
- NNS.caus, [13](#), [22](#), [28](#)
- NNS.cor, [14](#), [20](#), [22](#), [28](#)
- NNS.cor.hd, [15](#)
- NNS.dep, [7](#), [16](#), [20](#), [22](#)
- NNS.diff, [17](#)
- NNS.FSD, [18](#)
- NNS.FSD.uni, [18](#)
- NNS.norm, [19](#), [22](#)
- NNS.part, [20](#)
- NNS.reg, [6](#), [7](#), [20](#), [21](#), [27–29](#)
- NNS.SD.Efficient.Set, [24](#)
- NNS.seas, [25](#)
- NNS.SSD, [26](#)
- NNS.SSD.uni, [27](#)
- NNS.stack, [22](#), [27](#)
- NNS.term.matrix, [29](#)
- NNS.TSD, [30](#)
- NNS.TSD.uni, [31](#)
- plot3d, [15](#)
- PM.matrix, [31](#)
- sapply, [28](#)
- uniroot, [9](#), [10](#), [33](#)
- UPM, [32](#)
- UPM.VaR, [33](#)