Package 'NNS'

November 14, 2026

Type Package

Title Nonlinear Nonparametric Statistics

Version 14.1

Date 2026-11-14

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

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

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

License GPL-3

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

Depends R (>= 3.6.0)

Imports Rcpp, RcppParallel

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

VignetteBuilder knitr

LinkingTo Rcpp, RcppParallel

SystemRequirements GNU make

Config/testthat/edition 3

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation yes

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

Repository CRAN

Date/Publication 2026-11-14 16:00:10 UTC

Contents

Co.LPM . . . 3
Co.LPM_nD . . . 4
Co.UPM . . . 5
Co.UPM_nD . . . 6
D.LPM . . . 6
D.UPM . . . 7
DPM_nD . . . 8
dy.dx . . . 9
dy.d_ . . . 11
LPM . . . 13
LPM.ratio . . . 14
LPM.VaR . . . 15
NNS.ANOVA . . . 16
NNS.ARMA . . . 19
NNS.ARMA.optim . . . 21
NNS.boost . . . 24
NNS.caus . . . 27
NNS.CDF . . . 30
NNS.copula . . . 32
NNS.dep . . . 33
NNS.diff . . . 35
NNS.distance . . . 37
NNS.FSD . . . 37
NNS.FSD.uni . . . 38
NNS.gravity . . . 40
NNS.MC . . . 41
NNS.meboot . . . 42
NNS.mode . . . 46
NNS.moments . . . 47
NNS.norm . . . 48
NNS.part . . . 49
NNS.reg . . . 51
NNS.rescale . . . 56
NNS.SD.cluster . . . 57
NNS.SD.efficient.set . . . 59
NNS.seas . . . 60
NNS.SS . . . 62
NNS.SSD . . . 65
NNS.SSD.uni . . . 66

NNS.stack 67
NNS.TSD 72
NNS.TSD.uni 73
NNS.VAR 74
PM.matrix 77
UPM 78
UPM.ratio 79
UPM.VaR 80

Index **82**

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

Description

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

Usage

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

Arguments

`degree_lpm`	numeric; degree for x ("degree_x"). degree = 0 gives frequency, degree = 1 gives area.
`x`	numeric vector of observations.
`y`	numeric vector of the same length as x.
`target_x`	numeric vector; thresholds for x. There is no default: pass `NULL` to use `mean(x)`.
`target_y`	numeric vector; thresholds for y. There is no default: pass `NULL` to use `mean(y)`. `target_x` and `target_y` must be non-empty and the same length, or one of them length 1.
`degree_y`	numeric; optional degree for y. If omitted, ‘degree_lpm‘ is used for both x and y.

Value

Numeric vector of co-LPM values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

Co.LPM_nD *Co-Lower Partial Moment nD*

Description

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

Usage

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

Arguments

data	A numeric matrix with observations in rows and variables in columns.
target	A numeric vector, length equal to ncol(data).
degree	numeric; degree for lower deviations (0 = frequency, 1 = area).
norm	logical; if TRUE (default) divide by the sum of the co-lower, co-upper and divergent nD partial moments at the same degree and target ($\rightarrow$ [0,1]), otherwise return the raw moment. norm is ignored at degree = 0.

Value

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

`degree_upm`	numeric; degree for x ("degree_x"). degree = 0 gives frequency, degree = 1 gives area.
`x`	numeric vector of observations.
`y`	numeric vector of the same length as x.
`target_x`	numeric vector; thresholds for x. There is no default: pass `NULL` to use `mean(x)`.
`target_y`	numeric vector; thresholds for y. There is no default: pass `NULL` to use `mean(y)`. `target_x` and `target_y` must be non-empty and the same length, or one of them length 1.
`degree_y`	numeric; optional degree for y. If omitted, ‘degree_upm‘ is used for both x and y.

Value

Numeric vector of co-UPM values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

Co.UPM_nD *Co-Upper Partial Moment nD*

Description

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

Usage

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

Arguments

data	A numeric matrix with observations in rows and variables in columns.
target	A numeric vector, length equal to ncol(data).
degree	numeric; degree for upper deviations (0 = frequency, 1 = area).
norm	logical; if TRUE (default) divide by the sum of the co-lower, co-upper and divergent nD partial moments at the same degree and target ($\rightarrow$ [0,1]), otherwise return the raw moment. norm is ignored at degree = 0.

Value

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

Examples

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

## End(Not run)
```

D.LPM *Divergent-Lower Partial Moment*

Description

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

Usage

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

Arguments

degree_lpm	numeric; LPM degree = 0 gives frequency, = 1 gives area.
degree_upm	numeric; UPM degree = 0 gives frequency, = 1 gives area.
x	numeric vector of observations.
y	numeric vector of the same length as x.
target_x	numeric vector; thresholds for x. There is no default: pass NULL to use mean(x).
target_y	numeric vector; thresholds for y. There is no default: pass NULL to use mean(y). target_x and target_y must be non-empty and the same length, or one of them length 1.

Value

Numeric vector of divergent LPM values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

D.UPM *Divergent-Upper Partial Moment*

Description

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

Usage

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

Arguments

degree_lpm	numeric; LPM degree = 0 gives frequency, = 1 gives area.
degree_upm	numeric; UPM degree = 0 gives frequency, = 1 gives area.
x	numeric vector of observations.
y	numeric vector of the same length as x.
target_x	numeric vector; thresholds for x. There is no default: pass NULL to use mean(x).
target_y	numeric vector; thresholds for y. There is no default: pass NULL to use mean(y). target_x and target_y must be non-empty and the same length, or one of them length 1.

Value

Numeric vector of divergent UPM values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

DPM_nD *Divergent Partial Moment nD*

Description

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

Usage

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

Arguments

data	A numeric matrix with observations in rows and variables in columns.
target	A numeric vector, length equal to ncol(data).
degree	numeric; degree for upper deviations (0 = frequency, 1 = area).
norm	logical; if TRUE (default) divide by the sum of the co-lower, co-upper and divergent nD partial moments at the same degree and target ($\rightarrow$ [0,1]), otherwise return the raw moment. norm is ignored at degree = 0.

Value

Numeric; the n-dimensional divergent partial moment.

Examples

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

## End(Not run)
```

---

dy.dx *Partial Derivative dy/dx*

Description

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

Usage

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

Arguments

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

Details

`eval.point = "overall"` returns the mean of the `gradient` column of `NNS.reg(x, y)$Fitted.xy`. Each observation receives the slope of the unsmoothed regression segment that contains it, so the result is an observation-weighted average of segment slopes, not an average of smoothed derivatives. Very short segments between nearly coincident regression points can have extreme slopes; `mean(dy.dx(x, y, eval.point = x)$first.derivative)` averages the smoothed pointwise estimates instead.

Numeric evaluation points use central differences of smoothed NNS.reg estimates over five steps from 2 to `floor(sqrt(n))` typical spacings of `x`, clipped to the observed range. `second.derivative` is a second difference of those estimates, and second differences amplify noise: for `y = x^2` plus noise with an R-squared near 0.99 and n = 400, estimates of the true value 2 at three interior points had a root mean squared error of about 5, and about one in five had the wrong sign. Treat it as a descriptive diagnostic on noisy data.

Value

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

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

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

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

- Numeric values must be in matrix or data.frame form to be evaluated for each regressor, otherwise, a vector of points will evaluate only at the `wrt` regressor, averaging over the observed rows of the other regressors (at most `max(21, 10000 / k)` of them for `k` points, taken evenly through the data). See examples for use cases.
- Set to `(eval.points = "obs")` (default) to find the average partial derivative at every observation of the variable with respect to *for specific tuples of given observations.*
- Set to `(eval.points = "apd")` to find the average partial derivative at every observation of the variable with respect to *over the entire distribution of other regressors*, represented by their observed rows.
- Set to `(eval.points = "mean")` to find the partial derivative at the mean of value of every variable.
- Set to `(eval.points = "median")` to find the partial derivative at the median value of every variable.
- Set to `(eval.points = "last")` to find the partial derivative at the last observation of every value (relevant for time-series data).

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

Details

Derivatives are finite-difference summaries of the fitted NNS.reg surface, averaged over four local probability windows derived from degree-1 partial-moment quantiles. Window sizes depend on the number of observations; they are capped before constructing a local coordinate half-width. All evaluation points and perturbed coordinates must stay inside the observed coordinate ranges. Interior unequal spacings use polynomial finite-difference weights; support endpoints use one-sided three-point formulas. This avoids averaging artificial full-support extrapolation bands into a local derivative.

The coordinate-range check does not establish that every perturbed tuple lies in a densely observed part of the joint support. Sparse cells, neighbour changes, fitted-surface bias, and correlated predictors can affect the result, including at the boundary. These are numerical derivatives of an estimated surface, with no universal recovery or coverage guarantee. A two-valued coordinate returns its finite contrast divided by the coordinate difference and an undefined (NA) second derivative. Constant coordinates return NA. Mixed derivatives require two predictors and explicit matrix evaluation points.

Value

Returns column-wise matrix of wrt regressors:

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

Note

Binary regressors provide a finite contrast, not an identified continuous derivative between the two levels.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

## To find derivatives of y wrt 1st regressor for specific points of both regressors
dy.d_(B, y, wrt = 1, eval.points = t(c(.5, median(x_2))))
```

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

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


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

## Central 95% range across estimated partial derivatives (descriptive)
### Lower CI
LPM.VaR(.025, 0, apd[,1]$First)

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

## End(Not run)
```

LPM *Lower Partial Moment*

Description

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

Usage

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

Arguments

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

Value

LPM of variable

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

LPM.ratio *Lower Partial Moment Ratio*

Description

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

Usage

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

Arguments

degree	numeric; degree = 0 gives frequency (CDF), degree = 1 gives area.
target	numeric vector; threshold(s). There is no default: pass NULL to use mean(variable).
variable	numeric vector or data-frame column to evaluate.

Value

Numeric vector of standardized lower partial moments. For degree > 0, a target at which every observation equals the target (0/0) returns the degree = 0 ratio.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

## End(Not run)
```

LPM.VaR *LPM VaR*

Description

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

Usage

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

Arguments

percentile	numeric [0, 1]; The percentile for left-tail VaR.
degree	integer; (degree = 0) gives the interpolated empirical quantile (quantile(x, percentile, type = 7)), which on tied data need not be an exact inverse of LPM.ratio(0, ., x). Use it for probability statements, such as a VaR exceeded with a given frequency. (degree = 1) gives an area-weighted (severity-weighted) threshold, the percentile-expectile of x (Newey and Powell 1987): percentile of the area of deviations, not of the observations, lies below it, so it is not a quantile of x (for exponential data the degree-1 5% point has about a quarter of the observations below it). Higher degrees weight large deviations more.
x	a numeric vector.

Value

Returns the type-7 empirical quantile (degree = 0), whose strict or inclusive sample tail fraction may differ from percentile because of interpolation or ties, or the threshold with percentile of the directional deviation area below it (degree > 0).

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

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

## End(Not run)
```

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

Description

Compares control and treatment groups through benchmark-relative partial moments. The chosen mode compares means, medians, or a distribution profile with central and tail benchmarks, without selecting a parametric family.

Usage

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

Arguments

control	Numeric vector of control group observations
treatment	Numeric vector of treatment group observations
means.only	Logical; FALSE (default) uses full distribution analysis. Set TRUE for the central comparison only (means, or medians when medians = TRUE)
medians	Logical; FALSE (default) uses means. Set TRUE for median-based analysis
confidence.interval	Numeric [0,1]; confidence level for effect size bounds (e.g., 0.95)
tails	Character; specifies CI tail(s): "both", "left", or "right"
pairwise	logical; FALSE (default) Returns pairwise certainty tests when set to pairwise = TRUE.

plot	Logical; TRUE (default) generates distribution plot
robust	logical; FALSE (default) Two-sample (control, treatment) path only: draws 100 independent bootstrap resamples of each group (with replacement, from its own full sample), and returns / plots the robust central tendency (NNS.gravity) and the confidence.interval bounds of the resampled Certainty values. It is ignored when control is a matrix or list (multi-group, including pairwise = TRUE).

Details

The key output, Certainty, is a descriptive agreement index in $[0, 1]$ for the chosen target:

- means.only = TRUE, medians = FALSE: compares group means using degree-one deviation-area ratios at the pooled mean.
- means.only = TRUE, medians = TRUE: compares medians using the distance of 0.5 from each interval $[F(t-), F(t)]$ at the pooled median t. This interval contains 0.5 exactly when t is a median, including tied and discrete data.
- means.only = FALSE: adds the 12.5% and 25% lower and upper deviation-area benchmarks to the chosen central comparison. This examines distributional shape through a finite profile; agreement at these benchmarks does not prove equality of laws.

At a constant group's only value both deviation areas vanish. For agreement comparisons this satisfies the expectile balance at every level, so its area discrepancy is zero there. Returned CDF values retain the public partial-moment step convention; other benchmarks and nonconstant samples use the ordinary ratio discrepancy.

The raw score is 1 at maximal benchmark agreement. The returned score is multiplied by $((n_1 + n_2 - 2)/(n_1 + n_2))^2$, so the finite-sample maximum is below 1. Certainty is not a calibrated probability, a p-value, or statistical power, and has no universal significance cutoff. Equal population means need not produce the same score distribution when sample sizes or spreads differ. robust = TRUE describes sampling variability of the score; it does not generate a null reference distribution.

Random relabelings of the pooled observations provide a reference under identical distributions only when observations are exchangeable. They are not a general unequal-variance equal-means test. Any inferential calibration must match the chosen target and the sampling design; dependent observations require a suitable resampling design.

Effect-size bounds retain the conservative difference of the groups' separate bootstrap endpoints. They are not quantiles of the bootstrap difference and need not have exactly the requested nominal coverage. The independent-group bootstrap assumes independent observations within and between groups.

Value

Returns a list containing:

- Control: Mean/median of control group
- Treatment: Mean/median of treatment group
- Grand_Statistic: Mean/median of all observations pooled
- Control_CDF: CDF value at grand statistic (control)

- `Treatment_CDF`: CDF value at grand statistic (treatment)
- `Certainty`: Agreement index for the chosen central or distribution-profile comparison, with the finite-sample adjustment; not a calibrated probability.
- `Effect_Size_LB`: Lower bound of treatment effect (if confidence.interval requested)
- `Effect_Size_UB`: Upper bound of treatment effect (if confidence.interval requested)
- `Confidence_Level`: Confidence level used (if confidence.interval requested)

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

### Two variable analysis with no control variable
A <- cbind(x, y)
NNS.ANOVA(A)

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

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

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

## End(Not run)
```

NNS.ARMA *NNS ARMA*

Description

Autoregressive model incorporating nonlinear regressions of component series.

Usage

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

Arguments

variable	a numeric vector.
h	integer; 1 (default) Number of periods to forecast.
training.set	numeric; NULL (default) Sets the number of variable observations (variable[1 : training.set]) to monitor performance of forecast over in-sample range.
seasonal.factor	logical or integer(s); TRUE (default) Automatically selects the best seasonal lag from the seasonality test, NNS.seas's best.period without its permutation test. For a significance-filtered lag, run NNS.seas(variable, p.value = TRUE) first and pass its best.period (or periods) here; best.period is 1 when no period is significant. To use weighted average of all seasonal lags set to (seasonal.factor = FALSE). Otherwise, directly input known frequency integer lag to use, i.e. (seasonal.factor = 12) for monthly data. Multiple frequency integers can also be used, i.e. (seasonal.factor = c(12, 24, 36))
weights	numeric or "equal"; NULL (default) sets the weights of the seasonal.factor vector when specified as integers. If (weights = NULL) each seasonal.factor is weighted on its NNS.seas result and number of observations it contains, else

an "equal" weight is used. Numeric weights must have one value per seasonal lag and are treated as relative weights (rescaled to sum to 1). With (dynamic = TRUE) the lags are re-selected at every step, so only NULL or "equal" weights are accepted.

best.periods — integer; 1 (default) used in conjunction with (seasonal.factor = FALSE), uses the best.periods number of detected seasonal lags instead of ALL lags when (seasonal.factor = FALSE, best.periods = NULL).

modulo — integer(s); NULL (default) Adds the multiples of modulo on either side of each detected period (period 7 with modulo = 5 gives 5 and 10).

mod.only — logical; TRUE (default) Limits the number of seasonal periods returned to the specified modulo.

negative.values — logical; FALSE (default) If the variable can be negative, set to (negative.values = TRUE). If there are negative values within the training data variable[1 : training.set], negative.values will automatically be detected.

method — options: ("lin", "nonlin", "both", "means"); "nonlin" (default) To select the regression type of the component series, select (method = "both") where both linear and nonlinear estimates are generated. To use a nonlinear regression, set to (method = "nonlin"); to use a linear regression set to (method = "lin"). Means for each subset are returned with (method = "means").

dynamic — logical; FALSE (default) To update the seasonal factor with each forecast point, set to (dynamic = TRUE). The default is (dynamic = FALSE) to retain the original seasonal factor from the inputted variable for all ensuing h.

shrink — logical; FALSE (default) Ensembles forecasts with method = "means".

plot — logical; TRUE (default) Returns the plot of all periods exhibiting seasonality and the variable level reference in upper panel. Lower panel returns original data and forecast.

seasonal.plot — logical; TRUE (default) Adds the seasonality plot above the forecast. Will be set to FALSE if no seasonality is detected or seasonal.factor is set to an integer value.

pred.int — numeric [0, 1]; NULL (default) Plots and returns lower and upper bands around the forecasts. The forecast path itself is resampled with the maximum entropy bootstrap (NNS.MC over rho from -1 to 1), the bounds at each step are LPM.VaR and UPM.VaR of the replicates at probability (1 - pred.int) / 2, widened by the spread of the per-lag linear estimates; a one-point (h = 1) or constant path has no dispersion to resample, so its band is the path widened by that spread alone (zero width with a single seasonal lag); NNS.ARMA.optim gives intervals from validation errors. The band therefore describes the dispersion of the forecast path, not the uncertainty of the outcome at each horizon: it does not widen with the horizon, and it is not calibrated to contain future values with probability pred.int. NNS.ARMA.optim derives its interval from validation errors instead.

Value

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

Note

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

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

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

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

## End(Not run)
```

NNS.ARMA.optim *NNS ARMA Optimizer*

Description

Wrapper function for optimizing any combination of a given `seasonal.factor` vector in NNS.ARMA. The objective function `obj.fn` on a validation window (the last `length(variable) - training.set` observations) is used to determine the optimum across all NNS.ARMA methods.

Usage

```
NNS.ARMA.optim(
  variable,
  h = NULL,
  training.set = NULL,
  seasonal.factor,
```

```
  lin.only = FALSE,
  negative.values = FALSE,
  obj.fn = expression({
     .denom <- NNS::Co.LPM(1, predicted, actual, target_x =
   mean(predicted), target_y = mean(actual)) + NNS::Co.UPM(1, predicted, actual,
    target_x = mean(predicted), target_y = mean(actual))
     if (.denom == 0) Inf else
    mean((predicted - actual)^2)/.denom
 }),
  objective = "min",
  linear.approximation = TRUE,
  ncores = NULL,
  pred.int = 0.95,
  print.trace = TRUE,
  plot = FALSE,
  bias.correction = c("none", "split")
)
```

Arguments

variable	a numeric vector.
h	integer; NULL (default) Number of periods to forecast out of sample. If NULL, h = length(variable) - training.set.
training.set	integer; NULL (default) Sets the number of variable observations as the training set. See Note below for recommended uses.
seasonal.factor	integers; Multiple frequency integers considered for NNS.ARMA model, i.e. (seasonal.factor = c(12, 24, 36)).
lin.only	logical; FALSE (default) For fast optimization of the linear regression method. More robust than lin.only = TRUE.
negative.values	logical; FALSE (default) If the variable can be negative, set to (negative.values = TRUE). It will automatically select (negative.values = TRUE) if the training data variable[1 : training.set] has a negative value, and for the out-of-sample forecast if variable does.
obj.fn	expression; The default is the mean squared error divided by the degree-1 co-movement of predicted and actual, mean((predicted - actual)^2) / (Co.LPM(1, predicted, actual, mean(predicted), mean(actual)) + Co.UPM(1, predicted, actual, mean(predicted), mean(actual))) (Inf when the co-movement is 0), to be minimized. 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.
linear.approximation	logical; TRUE (default) Uses the best linear output from NNS.reg to generate a nonlinear and mixture regression for comparison. FALSE is a more exhaustive search over the objective space.

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

`pred.int` numeric [0, 1]; 0.95 (default) Returns the associated prediction intervals for the final estimate. Constructed from the returned model's internal validation residuals (actual - forecast), adding their LPM.VaR and UPM.VaR at `(1 - pred.int)/2` to the final estimates.

`print.trace` logical; `TRUE` (default) Prints current iteration information. Suggested as backup in case of error, best parameters to that point still known and copyable!

`plot` logical; `FALSE` (default)

`bias.correction`

`"none"` (default) returns the selected model's forecasts without an additive shift. `"split"` estimates a gravity bias from the first half of the validation errors and accepts it only if the objective improves on the later half; at least four validation observations are required. On rolling-origin forecasts of nine series (AirPassengers, UKgas, LakeHuron, a random walk, sunspot.month, nottem, USAccDeaths, co2 and Nile; six expanding origins, h = 12), the uncorrected forecasts had the lower mean MASE on eight of the nine series, both against `"split"` and against estimating and accepting the shift on the same window, a rule that is no longer offered. Model selection still uses the validation window, so these diagnostics are not independent out-of-sample performance estimates.

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
- `$obj.fn` the `obj.fn` value of the returned model's internal validation forecasts, after `$bias.shift`.
- `$method` the method identifying which NNS.ARMA method was used.
- `$shrink` whether to use the `shrink` parameter in NNS.ARMA.
- `$nns.regress` whether to smooth the variable via NNS.reg before forecasting.
- `$bias.shift` the accepted additive correction (0 when rejected or disabled). Add it to the forecast from NNS.ARMA with the returned model parameters, then apply the nonnegative support clamp if required.
- `$bias.validation` correction policy, estimated/applied shifts, chronological estimate and acceptance rows, and raw/adjusted acceptance scores.
- `$raw.results` forecasts before the additive bias correction.
- `$validation.forecast` the final internal validation forecasts, including correction and support clamp.
- `$errors` a vector of the returned model's internal validation errors (forecast - actual), before `$bias.shift`.
- `$results` a vector of length `h`.
- `$lower.pred.int` a vector of lower prediction intervals per forecast point.
- `$upper.pred.int` a vector of upper prediction intervals per forecast point.

Note

By default no bias shift is added. With `bias.correction = "split"` the shift is estimated on the first half of the validation window and accepted on the second. Base model selection still uses the internal validation window, so these tuning scores are not an independent evaluation of the whole forecasting procedure. Evaluate the complete procedure at fresh chronological origins.

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

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

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

## End(Not run)
```

NNS.boost *NNS Boost*

Description

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

Usage

```
NNS.boost(
  IVs.train,
  DV.train,
  IVs.test = NULL,
  type = NULL,
  depth = NULL,
  learner.trials = 100,
  epochs = NULL,
  CV.size = NULL,
  balance = FALSE,
  ts.test = NULL,
  threshold = NULL,
  obj.fn = expression(sum((predicted - actual)^2)),
  objective = "min",
  extreme = FALSE,
  features.only = FALSE,
  feature.importance = TRUE,
  pred.int = NULL,
  status = TRUE,
  seed = 123L,
  dist = NULL,
  folds = 5
)
```

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. Columns are matched to (IVs.train) by name when the two share the same predictor names, and positionally when they share no names at all (as in cbind(test.x_1, test.x_2) against cbind(x_1, x_2)). A CSV UTF-8 byte-order mark in the first header (for example, Male becoming X...Male) is repaired before matching names. Other partial overlaps, including misspelled predictors, are ambiguous and return an error.
type	NULL (default). To perform a classification of discrete integer classes from factor target variable (DV.train) with a base category of 1, set to (type = "CLASS"), else for continuous (DV.train) set to (type = NULL).
depth	options: (integer, NULL, "max"); (depth = NULL)(default) Specifies the order parameter in the NNS.reg routine, assigning a number of splits in the regressors, analogous to tree depth.
learner.trials	integer; 100 (default) Sets the number of trials to obtain an accuracy threshold level. If the number of all possible feature combinations is less than selected value, the minimum of the two values will be used.

epochs	integer; 2*length(DV.train) (default) Number of repeated holdout re-evaluations of the learner-trial feature subsets that pass the accuracy threshold.
CV.size	numeric [0, 1]; NULL (default) Sets the cross-validation size. Defaults to a random value between 0.2 and 0.33 for a random sampling of the training set.
balance	logical; FALSE (default) Uses both up and down sampling to balance the classes. type="CLASS" required.
ts.test	integer; NULL (default) Sets the length of the test set for time-series data; typically 2*h parameter value from NNS.ARMA or double known periods to forecast.
threshold	numeric [0, 1]; NULL (default) Probability supplied to LPM.VaR over the learner-trial objective distribution to determine the objective cutoff for keeping feature combinations. Defaults to 0.80 when objective = "max" and 0.20 when objective = "min". It is not a literal objective-score cutoff.
obj.fn	expression; expression(sum((predicted - actual)^2)) (default) Sum of squared errors is the default objective function. Any expression(...) using the specific terms predicted and actual can be used. Automatically selects an accuracy measure when (type = "CLASS").
objective	options: ("min", "max") "min" (default) Select whether to minimize or maximize the objective function obj.fn.
extreme	logical; FALSE (default) Sets the LPM.VaR probability to 1 (0) for maximization (minimization) objective, i.e. the most extreme learner-trial objective value becomes the cutoff. Overrides threshold.
features.only	logical; FALSE (default) Returns only the final feature loadings along with the final feature frequencies.
feature.importance	logical; TRUE (default) Draws a two-panel diagnostic: the learner-trial objective distribution with its LPM.VaR cutoff, and the frequency of features used in the final estimate.
pred.int	numeric [0,1]; NULL (default) Returns the associated prediction intervals for the final estimate.
status	logical; TRUE (default) Prints status update message in console.
seed	Optional integer random seed used for reproducible resampling, fold construction, and stochastic fitting steps. If 'NULL', the current random-number-generator state is used.
dist	options:(NULL, "NNS", "L1", "L2", "FACTOR") the method of distance calculation passed to delegated NNS.reg and NNS.stack calls. dist = NULL is the default and selects the native blended NNS distance; dist = "NNS" is an explicit alias for the default.
folds	integer; 5 (default) Number of cross-validation folds passed to the final NNS.stack call.

Value

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

and (for classification) the class labels `$class.levels`. Classification results are numeric: a factor or character `DV.train` yields integer class codes with a base category of 1 (label recoverable as `class.levels[results]`), and a numeric `DV.train` yields its original numeric class values.

Note

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

```
## Using 'iris' dataset with a stratified test set of ten rows per species
## (rows 141:150 alone are all virginica).
## Not run:
test.set <- c(41:50, 91:100, 141:150)
a <- NNS.boost(iris[-test.set, 1:4], iris[-test.set, 5],
IVs.test = iris[test.set, 1:4],
epochs = 100, learner.trials = 100,
type = "CLASS", depth = NULL, balance = TRUE)

## Test accuracy
mean(a$results == as.numeric(iris[test.set, 5]))

## Recover the labels
a$class.levels[a$results]

## End(Not run)
```

`NNS.caus` *NNS Causation*

Description

Returns the directional lagged-association diagnostic between two variables from observational data, as an index in [0, 1] for each direction (0.5 = no net direction).

Usage

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

Arguments

x	a numeric vector, matrix or data frame.
y	NULL (default) or a numeric vector with compatible dimensions to x.
factor.2.dummy	logical; FALSE (default) Automatically augments variable matrix with numerical dummy variables based on the levels of factors. Includes dependent variable y.
tau	options: ("cs", "ts", integer); 0 (default) Lag between cause and effect. (tau = [integer]) compares the lagged pairs (x[t - tau], y[t]) for x —> y and (y[t - tau], x[t]) for y —> x. (tau = 0) or (tau = "cs") compares the contemporaneous pairs (cross-sectional data). (tau = "ts") uses one lag for both directions, the mean of the first NNS.seas period at most sqrt(length(x)) of each series.
plot	logical; FALSE (default) Plots both lagged pairs on the copula scale with their indices.
p.value	logical; FALSE (default) If TRUE, runs a permutation test of the signed statistic C(x--->y) - C(y--->x).
nperm	integer; number of permutations to use when p.value = TRUE. Default 100.
permute	one of "auto" (default), "y", "x", "both" or "shift"; how the null distribution is built. "shift" circularly shifts y by random distinct offsets at least max(tau + 1, n / 10) steps from alignment, so the within-series ordering is retained, up to the wraparound boundary, while cross-series alignment changes. This does not condition on the effect's own past. When at most nperm offsets are admissible, each is used once and the reported nperm is their number. "y", "x" and "both" shuffle the values, an independence null for cross-sections. "auto" uses "shift" when the lag is positive (including tau = "ts") and "y" otherwise. The lag of tau = "ts" is resolved once on the observed series and kept for every permutation. For a matrix x each choice applies to every pair, with column i as x and column j as y; "auto" shuffles every column for cross-sections.
seed	optional integer seed for reproducibility of the permutation test.
conf.int	numeric; 0.95 (default) confidence level for the partial-moment based interval computed on the permutation null distribution.

Details

Following book chapter 14, each direction is evaluated on explicitly lagged pairs mapped to the copula scale with per-column ranks (the degree-0 lower partial moment), so the result is invariant to shifts, rescaling and any increasing transform of either variable (reversing a variable's order changes the copula, and so the index). For x —> y the raw score is

$$\tilde{C}_{x\to y} = \frac{1}{2}\left[P_{x\to y} + \max(\rho_{x\to y} - \rho_{y\to x}, 0)\right],$$

where $P_{x\to y}$ is the share of the deviation of `y[t]` resolved by conditioning on the position of `x[t - tau]`, the deviation being the degree-2 lower plus upper partial moment about the mean with the $m/(m-1)$ factor, used as a finite-sample normalization under the permutation reference, and $\rho_{x\to y}$ is the debiased directional dependence of `y[t]` on `x[t - tau]` from NNS.dep's cells (the lagged co-partial moments of the pair), averaged over the levels of the partition on `x[t - tau]`. NNS.dep reports the strongest level; the average avoids that maximization step. The neutral value is 0.5, but finite-sample behavior depends on the data and the null construction. The index is

$$C_{x\to y} = \frac{\tilde{C}_{x\to y} + \kappa}{\tilde{C}_{x\to y} + \tilde{C}_{y\to x} + 2\kappa}, \quad C_{y\to x} = 1 - C_{x\to y},$$

with a neutral pseudo-count $\kappa = 1/\sqrt{n-\tau}$, a regularization that shrinks weak directional evidence toward 0.5. It does not make the index's spread under independence constant: the spread grows with persistence. For independent pairs with n = 300 and tau = 20, the standard deviation of the index was about 0.08 for iid series, 0.14 for AR(1) series with coefficient 0.9 and 0.17 for random walks, and more than half of the persistent pairs fell more than 0.1 from 0.5. Distance from 0.5 alone is therefore not a significance criterion. An invertible relationship (e.g. `y = 2 * x + 1`) has no cross-sectional direction and returns 0.5; a lag can produce unequal directional association in that case. The index measures directional lagged association. It does not adjust for the response's own history, confounding variables, common drivers, or interventions. A common driver that reaches y later than x can favor x —> y without a causal effect. Circular-shift inference requires a null model invariant under the allowed shifts; stationarity and preserved autocorrelation alone do not establish this. Restricted offsets and boundary wraparound make this a finite resampling diagnostic, not an exact conditional-causality test. Shuffled-value inference instead requires exchangeability. The reported interval is an interval of the permutation null, not an uncertainty interval for an identified causal parameter.

Value

If `p.value = FALSE` returns a named vector `c("C(x--->y)", "C(y--->x)")` of directional association indices in [0, 1] that sum to 1: above 0.5 means the first lagged direction has the larger score, and 0.5 is no net direction under this diagnostic. If `p.value = TRUE` returns a list with components: * `causation`: the vector above. * `p.value`: a list with empirical two-sided and one-sided p-values (x_causes_y, y_causes_x) of the signed statistic `C(x--->y) - C(y--->x)`, its null distribution and observed value, the partial-moment interval of the null, and metadata (permute, nperm). For a matrix `x`, entry `[i, j]` is `C(column i ---> column j)`, with a 0 diagonal. If `p.value = TRUE` for a matrix, the function returns a list with components: * `causality`: the causation matrix. * `lower_CI`, `upper_CI`: matrices with the partial-moment interval of each index under the permutation null. * `p.value_matrix`: matrix of empirical two-sided p-values of `C(i ---> j) - C(j ---> i)`.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

## Lagged causation: y follows x by one period
x <- rnorm(500) ; y <- c(0, 0.8 * x[-500]) + 0.3 * rnorm(500)
NNS.caus(x, y, tau = 1)

## Causal matrix: entry [i, j] is C(column i ---> column j)
NNS.caus(iris, tau = 0)

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

## End(Not run)
```

NNS.CDF *NNS CDF*

Description

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

Usage

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

Arguments

variable	a numeric vector or data.frame of >= 2 variables for joint CDF.
degree	numeric; (degree = 0) (default) is frequency, the empirical distribution of variable; (degree = 1) is area, the share of deviation area below each point, which is not $P(X \leq t)$.
target	numeric; NULL (default) Must lie within support of each variable (one value per column for multivariate variable).

`type` options("CDF", "survival", "hazard", "cumulative hazard"); `"CDF"` (default) Selects type of function to return. For multivariate `variable` the joint survival is $P(X_1 > t_1, \ldots, X_d > t_d)$ at `degree = 0`, by inclusion-exclusion over the column subsets for $d \leq 3$ and as the share of observations above t in every column for larger d, and the joint upper partial moment Co.UPM_nD at `degree > 0`, normalised like the CDF. The multivariate cumulative hazard is the transform $-\log S(t)$ and requires `degree = 0`; it is not automatically an integral of a scalar multivariate hazard. Multivariate `type = "hazard"` is unsupported and raises an error: a joint density-over-survival or a coordinate-hazard definition requires a separate estimator, and dividing the joint CDF by survival does not estimate either. At `degree > 0` with two or more columns, each target's co-partial moment is normalised by its own total, so the "CDF" and "survival" values are generalised co-partial-moment ratios, not a joint distribution function: they can assign a negative probability to a rectangle. At `degree = 0`, and in one dimension at any degree, they are proper distribution functions; at positive degree the univariate induced law describes deviation-area balance rather than the frequency law of the original variable.

`plot` logical; plots CDF.

Value

Returns:

- `"Function"` a data.frame (class `NNS.data.frame`) containing the observations and the resulting function of the variable.
- `"target.value"` value from the `target` argument.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

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

## Area-weighted CDF (degree = 1): the share of deviation area below each point
NNS.CDF(x, 1)
```

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

NNS.CDF(A, 0)

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

Returns a multivariate dependence value [0,1]. Each co-partial moment is compared with its value under independence, the product of the marginal partial moments at `target`, so independent variables give a value near 0 for any target and marginal distributions. At a target with every variable's observations on one side, the normalized directional-region shares carry no contrast and the neutral value is 0. This target degeneracy is not evidence that the variables are independent. In general, 0 and 1 describe this target-specific summary, not global independence or deterministic dependence.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

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

## End(Not run)
```

`NNS.dep` *NNS Dependence*

Description

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

Usage

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

Arguments

`x`	a numeric vector, matrix or data frame.
`y`	`NULL` (default) or a numeric vector with compatible dimensions to `x`.

asym	logical; FALSE (default) returns the symmetric measures. If TRUE, returns the asymmetric (directional) measures: the partition is built on x only and the result measures the dependence of y on x (y as a function of x); swap the arguments for the other direction.
p.value	logical; FALSE (default) If TRUE (bivariate only), compares the results against 100 random permutations of y: the Dependence p-value is the upper-tail permutation p-value (1 + #(null >= observed)) / (1 + 100), the Correlation p-value is the two-sided permutation p-value min(1, 2 * min(1 + #(null <= observed), 1 + #(null >= observed)) / (1 + 100)), and the 95 percent intervals of the permutation null are returned (and plotted when print.map = TRUE).
print.map	logical; FALSE (default) Plots quadrant means, or p-value replicates.

Details

Dependence is a finite-resolution summary of the recursive partitions of NNS.part, which split each cell at the gravity of its observations (noise.reduction = "off"), not at its mean. At each cell, degree-0 (frequency) and degree-1 (area) concordance around the cell means are measured relative to a within-cell permutation reference and rescaled. Their cell-size-weighted averages are evaluated across partition levels. The reported Dependence is the strongest level, floored at zero; Correlation is a signed summary at that selected level and is bounded by Dependence in absolute value. With asym = FALSE, partitions on x, on y, and on joint quadrants are considered. With asym = TRUE, only partitions on x are considered, so the direction describes the resolution supplied by the conditioning variable rather than a causal effect.

The split depth, stopping rules, finite occupancy, and maximization across levels affect the score. A zero score does not prove independence, a noiseless nonlinear function need not score one, and finite independent samples can produce positive scores. The permutation calculation repeats the same selection rule, providing a reference for the selected statistic rather than removing every finite-sample effect from the score itself.

The degree-0 cell reference uses the hypergeometric distribution, with a normal approximation for cell sizes above 400. The degree-1 reference enumerates distinct arrangements only when at most 720 are required (all arrangements for up to six distinct values); larger cells use an approximation. These are properties of the cell calibration, separate from the 100 whole-sample permutations used by p.value = TRUE.

Power depends on the alternative. On noisy linear relationships in small samples, rank-based, distance-correlation or Hoeffding tests can reject more often than the p.value permutation test of NNS.dep; on non-monotone or heteroscedastic shapes such as a circle or y = x * e, the NNS.dep test can reject far more often.

Whole-sample shuffling assumes exchangeability of observations under the null. It is not an automatic test for serially dependent or clustered observations. The returned fields named 95% CIs are central intervals of the permutation null, not confidence intervals for a population dependence parameter. Inclusive-tail counts with the plus-one correction avoid zero Monte Carlo p-values; ties make the test conservative.

Value

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

Note

For asymmetrical (asym = TRUE) matrices, entry [i, j] is the directional dependence ([row variable i] —> [column variable j]), i.e. NNS.dep(x[, i], x[, j], asym = TRUE), the dependence of column j on row variable i.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

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

## End(Not run)
```

NNS.diff *NNS Numerical Differentiation*

Description

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

Usage

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

Arguments

f	an expression or call or a formula with no lhs.
point	numeric; Point to be evaluated for derivative of a given function f.
h	numeric [0, ...]; Initial step for secant projection. Defaults to (h = abs(point) * 0.1 + 0.01).
tol	numeric; Stopping tolerance. The search stops when the inferred h is below tol or when the bracket on the y-intercept fixes the derivative to within about tol * max(1, abs(derivative)). Defaults to (tol = 1e-10).
max.iter	integer; NULL (default) Maximum number of bisection iterations. NULL sets the limit to 100L. For noisy functions the bisection may stall before tol is reached; max.iter provides a hard upper bound.
digits	numeric; Sets the number of digits specification of the output. Defaults to (digits = 12).
print.trace	logical; FALSE (default) Displays each iteration, lower y-intercept, upper y-intercept and inferred h.
plot	logical; plots range, secant lines and y-intercept convergence.

Value

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

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

## End(Not run)
```

NNS.distance *NNS Distance*

Description

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

Usage

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

Arguments

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

Value

Returns sum of weighted distances.

NNS.FSD *NNS FSD Test*

Description

Compares empirical first-order stochastic dominance or degree-one area-ratio (Omega/expectile) dominance using partial moments.

Usage

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

Arguments

x	a numeric vector.
y	a numeric vector.
type	"discrete" (default) compares empirical CDFs for ordinary first-order stochastic dominance. The legacy option "continuous" compares degree-one ratios $L_1/(L_1+U_1)$, equivalently Omega/expectile order; it does not carry the ordinary FSD guarantee for every increasing utility.
plot	logical; TRUE (default) plots the FSD test.

Value

With `type = "discrete"`, returns `"X FSD Y"`, `"Y FSD X"`, or `"NO FSD EXISTS"`. With `type = "continuous"`, returns `"X OMEGA Y"`, `"Y OMEGA X"`, or `"NO OMEGA DOMINANCE EXISTS"`; the numerical area-order criterion is unchanged.

Note

The curves are compared on the merged sample points and, for TSD and continuous area order, also at their closed-form extrema between those points. Differences smaller than `64 * .Machine$double.eps * S^d` (`S` = largest absolute value in `x` and `y`, `d` = 1 for SSD and the mean condition, 2 for TSD; a ratio analogue for continuous area order) are treated as ties, so that rounding noise does not decide dominance. Discrete FSD compares exact empirical CDF values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

x	a numeric vector.
y	a numeric vector.
type	"discrete" (default) uses empirical FSD. At degree 1, "continuous" retains the degree-one area-ratio (Omega/expectile) ordering, which need not imply FSD or preference for every increasing utility. SSD and TSD are unaffected by type.

Value

Returns (1) if X dominates Y in the selected order (ordinary FSD or Omega/expectile order), else (0).

Note

The curves are compared on the merged sample points and, for TSD and continuous area order, also at their closed-form extrema between those points. Differences smaller than 64 * .Machine$double.eps * S^d (S = largest absolute value in x and y, d = 1 for SSD and the mean condition, 2 for TSD; a ratio analogue for continuous area order) are treated as ties, so that rounding noise does not decide dominance. Discrete FSD compares exact empirical CDF values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

## End(Not run)
```

NNS.gravity *NNS gravity*

Description

Central tendency averaging the median, the mean, the midpoint of the quartiles and a histogram mode; less sensitive to outliers than the mean.

Usage

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

Arguments

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

Details

$$(Q_2 + m + \bar{x} + (Q_1 + Q_3)/2)/4,$$

with the quartiles Q_1, Q_2, Q_3 of `quantile(x, type = 7)` and, as the mode m, the mean of the observations in the tallest bin of a histogram with bin width IQR / sqrt(n) and its two neighbouring bins (all observations when several bins are equally tall). Up to three values give the median. The mean term remains unbounded under one-point contamination: a single 1e6 among 99 zeros gives 2500, against 10000 for the mean. The other components can also change, so the one-quarter factor is an example, not a universal sensitivity bound. Its asymptotic replacement breakdown point is zero; it is not a high-breakdown estimator.

Value

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

Author(s)

Fred Viole, OVVO Financial Systems

Examples

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

## End(Not run)
```

NNS.MC *NNS Monte Carlo Sampling*

Description

Monte Carlo sampling from the maximum entropy bootstrap routine NNS.meboot, ensuring the replicates are sampled across the full [-1,1] range of rho (the target dependence of the detrended residuals, see NNS.meboot).

Usage

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

Arguments

x	vector of data.
reps	numeric; number of replicates to generate for each rho value, 30 default. The defaults give 201 rho values and 6030 replicates.
lower_rho	numeric [-1,1]; -1 default will set the from argument in seq(from, to, by).
upper_rho	numeric [-1,1]; 1 default will set the to argument in seq(from, to, by).
by	numeric; .01 default will set the by argument in seq(-1, 1, step).
exp	positive finite numeric; 1 default will exponentially weight maximum rho value if exp > 1. Shrinks values towards upper_rho, keeping them within [lower_rho, upper_rho].
type	options("spearman", "pearson", "NNScor", "NNSdep"); type = "spearman"(default) dependence metric desired.
drift	logical; drift = TRUE (default) preserves the drift of the original series.
target_drift	numerical; target_drift = NULL (default) Specifies the desired drift when drift = TRUE, i.e. a risk-free rate of return.

target_drift_scale
: numerical; instead of calculating a target_drift, provide a scalar to the existing drift when drift = TRUE.

xmin
: numeric; the lower limit for the left tail.

xmax
: numeric; the upper limit for the right tail.

...
: possible additional arguments to be passed to NNS.meboot.

Value

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

References

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

Examples

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

## End(Not run)
```

NNS.meboot *NNS meboot*

Description

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

Usage

```
NNS.meboot(
  x,
  reps = 999,
  rho = NULL,
  type = "spearman",
  drift = TRUE,
  target_drift = NULL,
  target_drift_scale = NULL,
  trim = 0.1,
  xmin = NULL,
  xmax = NULL,
  reachbnd = TRUE,
```

```
  expand.sd = TRUE,
  force.clt = TRUE,
  scl.adjustment = FALSE,
  sym = FALSE,
  elaps = FALSE,
  digits = 6,
  colsubj,
  coldata,
  coltimes,
  ...,
  expand.residual.sd = TRUE
)
```

Arguments

x	vector of data.
reps	numeric; number of replicates to generate.
rho	numeric [-1,1] (vectorized); A rho must be provided, otherwise a blank list will be returned. The dependence target is applied to each individual replicate (every replicate is mixed to the requested dependence with the original series); it is **not** applied to the ensemble. See Note.
type	options("spearman", "pearson", "NNScor", "NNSdep"); type = "spearman"(default) dependence metric desired.
drift	logical; drift = TRUE (default) preserves the drift of the original series. drift = FALSE removes it: replicates are a flat baseline at mean(x) plus the resampled residuals.
target_drift	numerical; target_drift = NULL (default) Specifies the desired drift when drift = TRUE, i.e. a risk-free rate of return. A drift other than the original is centred so the replicates keep mean(x).
target_drift_scale	numerical; instead of calculating a target_drift, provide a scalar to the existing drift when drift = TRUE.
trim	numeric [0,1]; The mean trimming proportion, defaults to trim = 0.1.
xmin	numeric; the lower limit for the left tail.
xmax	numeric; the upper limit for the right tail.
reachbnd	logical; If TRUE potentially reached bounds (xmin = smallest value - trimmed mean and xmax = largest value + trimmed mean) are given when the random draw happens to be equal to 0 and 1, respectively.
expand.sd	logical; If TRUE the standard deviation in the ensemble is expanded. See expand.sd in meboot::meboot. The expansion toward sd(x) is skipped when drift = FALSE, target_drift or target_drift_scale replaces the original drift, since sd(x) includes that drift. A target equal to the original drift up to rounding (relative tolerance sqrt(.Machine$double.eps), plus an allowance for rounding in the slope fit that grows with max(abs(x))) keeps the original reconstruction.

force.clt	logical; If TRUE the replicate means are assigned an iid normal-reference scale: the replicate means are spread as a normal sample centred on mean(x) with standard deviation sd(x) / sqrt(n), the standard error of the mean of the n observations, whatever the number of reps. See force.clt in meboot::meboot, whose code divides by the number of replicates instead (its comment says n). When drift = FALSE, target_drift or target_drift_scale replaces the original drift, the replicate means are centred and spread using the new baseline plus the original residuals instead of x.
scl.adjustment	logical; If TRUE scale adjustment is performed to ensure that the population variance of the resampled residuals equals the variance of the detrended data.
sym	logical; If TRUE an adjustment is performed to ensure that the ME density is symmetric.
elaps	logical; If TRUE elapsed time during computations is displayed.
digits	integer; 6 (default) number of digits to round output to.
colsubj	numeric; the column in x that contains the individual index. It is ignored if the input data x is not a pdata.frame object.
coldata	numeric; the column in x that contains the data of the variable to create the ensemble. It is ignored if the input data x is not a pdata.frame object.
coltimes	numeric; an optional argument indicating the column that contains the times at which the observations for each individual are observed. It is ignored if the input data x is not a pdata.frame object.
...	possible argument fiv to be passed to expand.sd.
expand.residual.sd	logical; TRUE (default) retains the residual-space SD expansion before trend reconstruction. Set to FALSE to leave the generated residual scales unexpanded; this is separate from the later level-space expand.sd switch.

Value

Returns the following row names in a matrix:

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

- ordxx ordered x values.
- kappa scale adjustment to the variance of ME density.
- elaps elapsed time.

Note

These are dependence-targeted synthetic scenarios. Zero Spearman or Pearson correlation does not imply independence; interior-rho mixtures can retain nonlinear dependence and change the residual marginal distribution. Matching ranks or path moments does not by itself calibrate sampling confidence intervals. The `force.clt` scale `sd(x)/sqrt(n)` is an iid finite-variance reference, not a long-run standard error under serial dependence, and does not calibrate uncertainty of other statistics.

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

The `rho` dependence alignment is calibrated on each individual `replicate`: every replicate's *detrended residuals* are mixed so that their dependence on the original series' detrended residuals matches `rho`, in the metric implied by `type`; the trend is added back afterwards. If the detrended residuals are all exactly zero (a constant or exactly linear series) there is nothing to align and `rho` has no effect. For a trending series the replicate levels therefore stay strongly correlated with the original even at `rho = -1` (e.g. AirPassengers: mean Spearman 0.81 at `rho = -1`). Assess a result in that **same** metric – a `"NNSdep"` target with NNS.dep`$Dependence` (unsigned) and a `"spearman"`/`"pearson"` target with rank/linear correlation – because a signed correlation taken on an unsigned `"NNSdep"` target is not comparable to `rho` (it can even read negative while the dependence target is met). Separately, the `ensemble` is the per-observation mean of the replicates – a central summary, not a single series carrying the target dependence – so a rank or linear correlation computed directly on the `ensemble` tends to read higher than the per-replicate `rho` (averaging amplifies the shared order). Verify `rho` on the `replicates`, in the metric implied by `type`, rather than on the `ensemble`.

References

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

Examples

```
## Not run:
# Target zero Spearman correlation of each replicate's detrended residuals
boots <- NNS.meboot(AirPassengers, reps = 100, rho = 0, drift = FALSE,
  expand.sd = FALSE, force.clt = FALSE)
```

```
# Inspect individual residual correlations, not the ensemble's level correlation
detrend <- function(z) residuals(lm(as.numeric(z) ~ seq_along(z)))
apply(boots["replicates",]$replicates, 2, function(z)
  cor(z, detrend(AirPassengers), method = "spearman"))

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

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

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

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

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

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

## End(Not run)
```

NNS.mode *NNS mode*

Description

Mode of a distribution, either continuous or discrete.

Usage

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

Arguments

x	vector of data.
discrete	logical; FALSE (default) for continuous distributions, TRUE for discrete distributions.
multi	logical; TRUE (default) returns multiple mode values. FALSE returns the highest peak; among equally high peaks, the one nearest NNS.gravity of the data (then the lower value), so the result is always one of the modes.

Details

The continuous mode finds the peaks of a smoothed histogram with bin width IQR / sqrt(n), peaks in the first and last bins included, and centres each peak on the mean of the observations within three bins of it, so a peak on a single value returns that value. When every gap between consecutive distinct values is wider than a bin, the data are treated as discrete-valued: the modes are the values more frequent than their neighbouring values, a run of equally frequent values counting together. `discrete = TRUE` instead rounds to integers and returns the most frequent integers.

Value

Returns the mode of the distribution: a numeric vector of all modes when `multi = TRUE` (default), otherwise a single numeric value.

Author(s)

Fred Viole, OVVO Financial Systems

Examples

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

## End(Not run)
```

NNS.moments *NNS moments*

Description

This function returns the first 4 moments of the distribution.

Usage

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

Arguments

`x`	a numeric vector.
`population`	logical; `TRUE` (default) returns the population (unadjusted) statistics. `FALSE` returns the sample-adjusted statistics (variance with `n - 1`, sample skewness and sample excess kurtosis).

Value

Returns:

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

## End(Not run)
```

NNS.norm *NNS Normalization*

Description

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

Usage

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

Arguments

X	a numeric matrix or data frame, or a list.
linear	logical; FALSE (default) performs the nonlinear scaling normalization (each variable's mean-ratio scale is weighted by its correlation / NNS.dep dependence with the other variables). TRUE performs a linear scaling normalization, resulting in equal means for all variables. Both scale by ratios of means, so they assume variables with positive means; a zero mean is replaced by 1e-10 (giving extreme scale factors).

chart.type	options: ("l", "b"); NULL (default). Set (chart.type = "l") for line, (chart.type = "b") for boxplot.
location	Sets the legend location within the plot, per the x and y co-ordinates used in base graphics legend.

Value

Returns a data.frame of normalized values.

Note

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

### Normalize list of unequal vector lengths

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

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

## End(Not run)
```

NNS.part *NNS Partition Map*

Description

Creates partitions based on partial-moment quadrant centroids. Numeric orders use the compiled recursive partitioner. ‘order = "max"‘ returns the maximum observable partition representation without passing an invalid integer to C++.

Usage

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

Arguments

x	Numeric vector.
y	Numeric vector of the same length as x.
Voronoi	Logical; draw the partition map.
type	NULL or "XONLY".
order	NULL, a positive integer, or "max".
obs.req	Nonnegative integer minimum-observation stopping control.
min.obs.stop	Logical stopping control.
noise.reduction	One of "mean", "median", "mode", "mode_class", or "off".

Details

For a numeric `order`, each cell is split at a centre per coordinate, set by `noise.reduction`: `"mean"` and `"median"` use the mean and median, `"mode"` uses the continuous `NNS.mode(multi = FALSE)`, `"mode_class"` uses NNS.gravity for x and `NNS.mode(discrete = TRUE)` for y, and `"off"` uses NNS.gravity. Regression points are the same centres of each `prior.quadrant`. `order = "max"` makes no splits: every observation is its own regression point, or with `type = "XONLY"` each distinct x gets the y centre of its observations. Quadrant labels append one digit per order. With split centres `cx` and `cy`, the digit for the default joint partition (`type = NULL`) is `1 + (x <= cx) + 2 * (y <= cy)`: 1 = upper-right, 2 = upper-left, 3 = lower-right, 4 = lower-left. For `type = "XONLY"` the digit is 1 for `x <= cx` and 2 for `x > cx`.

Value

A list containing ‘order‘, ‘dt‘, and ‘regression.points‘.

- `order`: the order actually completed. It can be smaller than the requested `order` when `obs.req` / `min.obs.stop` stop the recursion (e.g. `order = 3` returns 2 for `n = 200` with the default `obs.req = 8`).
- `dt`: the observations with their final `quadrant` and `prior.quadrant` labels.
- `regression.points`: for a numeric `order`, one (x, y) centroid per `prior.quadrant`, i.e. per cell of the previous order: `order = 1` gives 4 final quadrants but a single regression point (`"q"`), `order = 2` gives up to 16 quadrants and 4 points.

NNS.reg *NNS Regression*

Description

Generates a nonlinear regression based on partial moment quadrant means.

Usage

```
NNS.reg(
  x,
  y,
  factor.2.dummy = TRUE,
  order = NULL,
  dim.red.method = NULL,
  tau = NULL,
  type = NULL,
  point.est = NULL,
  location = "top",
  return.values = TRUE,
  plot = TRUE,
  plot.regions = FALSE,
  residual.plot = TRUE,
  confidence.interval = NULL,
  threshold = 0,
  n.best = NULL,
  smooth = FALSE,
  noise.reduction = "off",
  dist = NULL,
  ncores = NULL,
  point.only = FALSE,
  multivariate.call = FALSE
)
```

Arguments

x	a vector, matrix or data frame of variables of numeric or factor data types.
y	a numeric or factor vector with compatible dimensions to x.
factor.2.dummy	logical; TRUE (default) Automatically augments variable matrix with numerical dummy variables based on the levels of factors.
order	integer; Controls the number of partial moment quadrant means. Users are encouraged to try different (order = ...) integer settings with (noise.reduction = "off"). (order = "max") retains distinct predictor locations. A perfect training fit requires compatible responses at tied locations and interpolation at those locations; duplicate predictors with different responses cannot all be interpolated.

`dim.red.method` options: ("cor", "NNS.dep", "NNS.caus", "all", "equal", `numeric vector`, NULL) method for determining synthetic X* coefficients (per Dana and Dawes (2004)). Selection of a method automatically engages the dimension reduction regression. The default is `NULL` for full multivariate regression. `(dim.red.method = "NNS.dep")` uses NNS.dep for nonlinear dependence weights, while `(dim.red.method = "NNS.caus")` uses NNS.caus for directional association weights. `(dim.red.method = "cor")` uses Spearman rank correlation for weights. `(dim.red.method = "all")` averages all methods for further feature engineering. `(dim.red.method = "equal")` uses unit weights. Alternatively, user can specify a numeric vector of coefficients. Named vectors must exactly match the encoded predictor columns and are matched by name; unnamed vectors follow encoded column order.

`tau` options("ts", NULL); `NULL`(default) To be used in conjunction with `(dim.red.method = "NNS.caus")` or `(dim.red.method = "all")`. If the regression is using time-series data, set `(tau = "ts")` to use the time-series lag convention.

`type` `NULL` (default). To perform a classification, set to `(type = "CLASS")`. Like a logistic regression, it is not necessary for target variable of two classes e.g. [0, 1]: a numeric response with exactly the values 0 and 1, or a factor, character or logical response, is classified automatically. Use `type = "REG"` to fit a numeric response as regression, including a two-valued response, with one or more predictors. `(type = "XONLY")` is the single-predictor regression alias and keeps the default x-only partition.

`point.est` a numeric or factor vector with compatible dimensions to `x`. Returns the fitted value `y.hat` for any value of `x`. Columns are matched to `x` by name when the two share the same predictor names, and positionally when they share no names at all (as in `cbind(test.x_1, test.x_2)` against `cbind(x_1, x_2)`). A CSV UTF-8 byte-order mark in the first header (for example, `Male` becoming `X...Male`) is repaired before matching names. Other partial overlaps, including misspelled predictors, are ambiguous and return an error.

`location` Sets the legend location within the plot, per the `x` and `y` co-ordinates used in base graphics legend.

`return.values` logical; `TRUE` (default), set to `FALSE` in order to only display a regression plot and call values as needed.

`plot` logical; `TRUE` (default) To plot regression.

`plot.regions` logical; `FALSE` (default). Generates 3d regions associated with each regression point for multivariate regressions. Note, adds significant time to routine.

`residual.plot` logical; `TRUE` (default) To plot `y.hat` and `Y`.

`confidence.interval` numeric [0, 1]; `NULL` (default) Plots the associated confidence interval with the estimate and adds `conf.int.pos` / `conf.int.neg` to `Fitted.xy`. The intervals are global: they are built from the type-8 quantiles of the in-sample residuals (no per-segment standard errors are returned). Also applies the same level for the prediction intervals.

`threshold` numeric [0, 1]; `(threshold = 0)` (default) Sets the threshold for dimension reduction of independent variables when `(dim.red.method)` is not `NULL`. Predictors whose coefficient magnitude is below `threshold` are dropped; `NULL` or `NA`

means 0. For classification, a vector with one cutoff per class (named by class, or in class order) applies each cutoff to that class's index, as NNS.stack can report in NNS.dim.red.threshold.

n.best integer; NULL (default) Sets the number of nearest regression points to use in weighting for multivariate regression at floor((1 - D) * sqrt(nrow(x))) (at least 1, at most the number of regression points), where D averages the copula summaries with response y and -y so the smoothing choice is unchanged by response reflection. (n.best = "all") will select and weight all generated regression points. Analogous to k in a k Nearest Neighbors algorithm. Different values of n.best are tested using cross-validation in NNS.stack. NA selects the default, as NULL does.

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

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

dist options:(NULL, "NNS", "L1", "L2", "FACTOR") the method of distance calculation. dist = NULL is the default and selects the native blended NNS distance, $sum(abs(z) + z^2)$, over range-normalized coordinates. dist = "NNS" is an explicit alias for the default. dist = "L2" selects Euclidean distance; dist = "L1" selects Manhattan distance; dist = "FACTOR" uses a frequency.

ncores integer; value specifying the number of cores to be used in the parallelized procedure. If NULL (default), the number of cores to be used is equal to the number of cores of the machine - 1 for multivariate regression predictions large enough to repay them, and one otherwise. The estimates do not depend on the number of cores.

point.only Internal argument for abbreviated output.

multivariate.call
Internal argument for multivariate regressions.

Details

For regression, R2 measures association between fitted and observed responses: it is $cor(y, \hat{y})^2$, with the package's finite constant-vector convention. It is not $1 - SSE/SST$; a biased or incorrectly scaled prediction can have R2 = 1. Evaluate held-out prediction errors for predictive accuracy.

The default partition order combines a finite-partition dependence score with the average of the duplicated-predictor copula score under both response orientations. This complexity heuristic is invariant to reflecting the response; the underlying directional dependence and copula definitions are unchanged. A supplied order bypasses this heuristic.

Univariate regression linearly interpolates the selected regression points and extends the first or last segment beyond their range (unless a smoothing spline is requested). Endpoint slopes can be unstable with sparse or noisy boundary points. Multivariate distance weighting is a weighted

average of regression-point responses, so it does not automatically continue a trend outside the training support. Neither construction supplies a general extrapolation guarantee. With the default `n.best = NULL`, purely categorical predictors use exact observed joint cells, and an unobserved joint combination uses its nearest cells. An explicit `n.best` weights that many nearest cells, as for numeric predictors. A declared factor level with no training rows is accepted in `point.est` and is placed by the same distance rule. A coordinate with exactly two observed values retains both terminal cells, including a dummy column in a mixed design. Other finite-order numeric coordinates retain interval cells; `order = "max"` uses exact joint support.

Classification forms an indicator for each observed class, retains the empirical class frequencies in each predictor cell, and interpolates or distance-weights those probability vectors. Class labels never enter numeric interpolation. `Point.probabilities`, `Fitted.probabilities`, and `RPM.probabilities` expose the corresponding vectors. Tied class decisions follow first appearance in the training response. Factor and character responses retain numeric output codes with `class.levels`; numeric responses retain their supplied values. For dimension reduction, one signed NNS index is fitted per class indicator, and the resulting probabilities are normalized across classes; `equation` is then a list indexed by class. `confidence.interval` returns `Prediction.set`, the smallest leading set of classes attaining the requested estimated probability mass. These are descriptive probability sets, not calibrated coverage guarantees; nominal classes have no numeric prediction interval.

Value

UNIVARIATE REGRESSION RETURNS THE FOLLOWING VALUES:

- `"R2"` is squared Pearson correlation between fitted and observed responses (Racine–Hastie goodness of fit), or fitted classification accuracy;
- `"SE"` returns the overall standard error of the estimate between `y` and `y.hat`;
- `"Prediction.Accuracy"` returns the proportion of fitted class decisions equal to the observed class;
- `"derivative"` for the coefficient of the `x` and its applicable range;
- `"Point.est"` for the predicted value generated;
- `"pred.int"` lower and upper prediction intervals for the `"Point.est"` returned using the `"confidence.interval"` provided;
- `"regression.points"` provides the points used in the regression equation for the given order of partitions;
- `"Fitted.xy"` returns a `data.frame` of `x`, `y`, `y.hat`, `resid`, `NNS.ID`, `gradient`;

MULTIVARIATE REGRESSION RETURNS THE FOLLOWING VALUES:

- `"R2"` is squared Pearson correlation between fitted and observed responses (Racine–Hastie goodness of fit), or fitted classification accuracy;
- `"equation"` returns the numerator of the synthetic X* dimension reduction equation as a `data.frame` consisting of regressor and its coefficient. Denominator is the number of non-zero coefficients (`abs(coefficient) > 0`), returned in last row of `equation` `data.frame`.
- `"x.star"` returns the synthetic X* as a vector;
- `"rhs.partitions"` returns the partition points for each regressor `x`;
- `"RPM"` provides the Regression Point Matrix, the points for each `x` used in the regression equation for the given order of partitions;

- "Point.est" returns the predicted value generated;
- "pred.int" lower and upper prediction intervals for the "Point.est" returned using the "confidence.interval" provided;
- "Fitted.xy" returns a data.frame of the regressors, y, y.hat, NNS.ID and residuals for full multivariate regression, or of the synthetic x, y, y.hat, NNS.ID, gradient and residuals for dimension reduction regression. confidence.interval adds conf.int.neg and conf.int.pos.

Note

- Please ensure point.est is of compatible dimensions to x, error message will ensue if not compatible.
- Like a logistic regression, the (type = "CLASS") setting is not necessary for target variable of two classes e.g. [0, 1]. For other numeric responses set type = "CLASS"; class values can be any distinct numbers.
- For low signal:noise instances, increasing the dimension may yield better results using NNS.stack(cbind(x,x), y, method = 1, ...).

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

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

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

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

Examples

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

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

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


## For Multiple Regression:
```

```
x <- cbind(rnorm(100), rnorm(100), rnorm(100)) ; y <- rnorm(100)
NNS.reg(x, y, point.est = c(.25, .5, .75))

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

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

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

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

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

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

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

## End(Not run)
```

`NNS.rescale` *NNS rescale*

Description

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

Usage

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

Arguments

x	numeric vector; data to rescale (e.g., terminal prices for risk-neutral method).
a	numeric; defines the scaling target: - For `method = "minmax"`: the lower limit of the output range (e.g., 5 to scale to [5, b]). - For `method = "riskneutral"`: the initial price \(S_0 \) (must be positive, e.g., 100), used to set the target mean.
b	numeric; defines the scaling range or rate: - For `method = "minmax"`: the upper limit of the output range (e.g., 10 to scale to [a, 10]). - For `method = "riskneutral"`: the risk-free rate \(r \) (e.g., 0.05), used with \(T \) to adjust the mean.

method	character; scaling method: "minmax" (default) for min-max scaling, or "riskneutral" for risk-neutral adjustment.
T	numeric; time to maturity in years (required for method = "riskneutral", ignored otherwise; e.g., 1). Default is NULL.
type	character; for method = "riskneutral": "Terminal" (default) or "Discounted" (mean = \(S_0 \)).

Value

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

Author(s)

Fred Viole, OVVO Financial Systems

Examples

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

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

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

## End(Not run)
```

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

Description

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

Usage

```
NNS.SD.cluster(
  data,
  degree = 1,
  type = "discrete",
```

```
  min_cluster = 1,
  dendrogram = FALSE
)
```

Arguments

data	A numeric matrix or data frame of variables to be clustered.
degree	Numeric options: (1, 2, 3). Degree of stochastic dominance test.
type	At degree 1, "discrete" (default) uses empirical FSD and "continuous" uses the degree-one area-ratio (Omega/expectile) ordering. The latter does not imply ordinary FSD. SSD and TSD are unaffected by type.
min_cluster	Integer. Peeling stops once at most min_cluster variables remain; those form the final cluster, and a final cluster smaller than min_cluster is merged into the previous one. SD-efficient sets peeled before that are kept whatever their size.
dendrogram	Logical; FALSE (default). If TRUE, a dendrogram is produced based on a simple "distance" measure between clusters.

Details

The function applies NNS.SD.efficient.set iteratively, peeling off the SD-efficient set at each step (whatever its size) while more than min_cluster variables remain, or until no more subsets can be extracted. Variables in each SD-efficient set form a cluster. The remaining variables form the final cluster, which is merged into the previous cluster if it has fewer than min_cluster members.

Value

A list with the following components:

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

```
A <- cbind(x, y, z)

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

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

## End(Not run)
```

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

Description

Determines the set of stochastic dominant variables for various degrees.

Usage

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

Arguments

`x`	a numeric matrix or data frame.
`degree`	numeric options: (1, 2, 3); Degree of stochastic dominance test from (1, 2 or 3).
`type`	`"discrete"` (default) uses empirical FSD. At degree 1, `"continuous"` retains the degree-one area-ratio (Omega/expectile) ordering, which need not imply FSD or preference for every increasing utility. SSD and TSD are unaffected by `type`.
`status`	logical; `TRUE` (default) Prints status update message in console.

Value

Returns set of stochastic dominant variable names.

Note

The curves are compared on the merged sample points and, for TSD and continuous area order, also at their closed-form extrema between those points. Differences smaller than `64 * .Machine$double.eps * S^d` (`S` = largest absolute value in `x` and `y`, `d` = 1 for SSD and the mean condition, 2 for TSD; a ratio analogue for continuous area order) are treated as ties, so that rounding noise does not decide dominance. Discrete FSD compares exact empirical CDF values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

Examples

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

## End(Not run)
```

`NNS.seas` *NNS Seasonality Test*

Description

Seasonality test based on the coefficient of variation for the variable and lagged component series. A linear trend is removed first (`detrend = TRUE`), keeping the mean of the variable, so that a trend does not mask the period. For each lag `m`, the variable is split into its `m` component series `variable[seq(j, n, m)]`, `j = 1, ..., m`, and lag `m` is reported when the length-weighted mean of the component coefficients of variation is below the coefficient of variation of the variable. Dispersion is the Gini mean difference, the degree-1 lower plus upper partial moment about each observation in turn, which is unbiased for any distribution at any component length. The coefficient of variation is used when the (detrended) variable stays on one side of zero; otherwise dispersions are compared directly. Lags are scanned up to `max.period`, by default `n / 5`, so every component series has at least 5 observations. Periods added through `modulo` are kept up to `n / 2`. On a series without seasonality about half of the lags pass by chance, so treat the result as candidate periods and validate them, for example with NNS.ARMA.optim, or set `p.value = TRUE`. A result of 1 signifies no seasonality present. A multiple of the true period scores as well in expectation, and its shorter components make its score noisier, so the lowest score often falls on a multiple. `best.period` is therefore the fundamental period: the smallest period that divides the lowest-scoring lag, or nearly divides it for a non-integer period, and whose score exceeds the lowest by at most two jackknife standard errors of their difference, from paired delete-one scores. It is listed first in `all.periods` and `periods`, followed by the other periods in order of score.

Usage

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

```
  max.period = NULL,
  p.value = FALSE,
  nperm = 100L,
  dispersion = c("auto", "absolute")
)
```

Arguments

variable	a numeric vector.
modulo	integer(s); NULL (default) Adds the multiples of modulo on either side of each detected period (period 7 with modulo = 5 gives 5 and 10).
mod.only	logical; TRUE (default) Limits the number of seasonal periods returned to the specified modulo.
plot	logical; TRUE (default) Returns the plot of all periods exhibiting seasonality and the variable level reference.
detrend	logical; TRUE (default) removes the variable's linear trend, keeping its mean, before the lags are scored; an exactly linear variable is then constant and has no seasonal period.
max.period	integer; NULL (default) scans lags up to n / 5. A smaller value caps the scan, whose cost grows with n times the number of lags.
p.value	logical; FALSE (default). If TRUE, each period's dispersion ratio is compared with the best ratio over all lags in nperm shuffled copies of the (detrended) variable, which remove the original temporal ordering. NNS.ARMA's automatic lag selection does not run this test; for a significance-filtered lag, pass NNS.seas(variable, p.value = TRUE)$best.period (or $periods) to its seasonal.factor. all.periods gains a p.value column, and periods and best.period keep only the periods with p.value <= 0.05 (1 if none); each copy contributes its best lag, accounting for scanning multiple lags under an exchangeable null. A serially dependent nonseasonal series need not satisfy that null. Uses the random number generator.
nperm	integer; 100 (default) number of shuffled copies when p.value = TRUE.
dispersion	"auto" (default) retains relative CV scoring on one-sided data and absolute Gini dispersion otherwise. "absolute" always compares Gini dispersions, by centering the series first; use this translation-invariant option for interval-scale data such as temperature.

Value

Returns a matrix of all periods exhibiting less coefficient of variation than the variable with "all.periods"; and the fundamental period with "best.period" (see Description); as well as a vector of "periods" for easy call into NNS.ARMA.optim. If no seasonality is detected, all.periods holds a single row with Period = 1 and a coefficient of variation equal to the variable's (Period = 0 for fewer than 5 observations). The coefficients of variation are those of the detrended variable when detrend = TRUE.

Note

The historical column names `Coefficient.of.Variation` and `Variable.Coefficient.of.Variation` are retained in absolute mode; their values then contain absolute Gini dispersions.

`best.period` need not be the lowest-scoring lag: `with(all.periods, Period[which.min(Coefficient.of.Variation)]` returns that lag. The jackknife behind the two-standard-error rule treats observations as exchangeable and does not adjust for serial dependence. The minimum five observations per component is a reliability default; `modulo` can supply longer domain-informed candidates. Shuffling removes serial dependence as well as seasonality, so the optional permutation comparison is not calibrated against every dependent nonseasonal process.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

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

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

## Keep only periods that beat shuffled copies of the series:
NNS.seas(x, plot = FALSE, p.value = TRUE)$best.period

## End(Not run)
```

NNS.SS *NNS Stochastic Superiority*

Description

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

Usage

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

Arguments

x	a numeric vector.
y	a numeric vector.
confidence.interval	logical; FALSE (default) returns only the empirical stochastic superiority measures. Set to TRUE to compute bootstrap confidence intervals for p_star.
reps	numeric; number of maximum entropy bootstrap replicates used when confidence.interval = TRUE. Default is 999.
ci	numeric in $(0, 1)$; confidence level used for the bootstrap interval when confidence.interval = TRUE. Default is 0.95.
rho	numeric; dependence target passed to NNS.meboot. Default is 1.

Details

NNS.SS returns:

$$P(X > Y),$$

the tie probability

$$P(X = Y),$$

and the tie-adjusted stochastic superiority measure

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

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

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

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

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

Value

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

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

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

`p_star` tie-adjusted stochastic superiority probability.

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

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

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

`p_star` tie-adjusted stochastic superiority probability.

`lower` lower confidence bound for `p_star`.

`upper` upper confidence bound for `p_star`.

`ci` confidence level used.

`reps` number of bootstrap replicates used.

`boot_vals` bootstrap replicate values of `p_star`.

Note

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

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

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

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

## End(Not run)
```

NNS.SSD *NNS SSD Test*

Description

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

Usage

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

Arguments

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

Value

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

Note

The curves are compared on the merged sample points and, for TSD and continuous FSD, also at their closed-form extrema between those points. Differences smaller than 64 * .Machine$double.eps * S^d (S = largest absolute value in x and y, d = 1 for SSD and the mean condition, 2 for TSD; a ratio analogue for continuous FSD) are treated as ties, so that rounding noise does not decide dominance. Discrete FSD compares exact empirical CDF values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

x	a numeric vector.
y	a numeric vector.

Value

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

Note

The curves are compared on the merged sample points and, for TSD and continuous area order, also at their closed-form extrema between those points. Differences smaller than `64 * .Machine$double.eps * S^d` (`S` = largest absolute value in `x` and `y`, `d` = 1 for SSD and the mean condition, 2 for TSD; a ratio analogue for continuous area order) are treated as ties, so that rounding noise does not decide dominance. Discrete FSD compares exact empirical CDF values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

## End(Not run)
```

`NNS.stack` *NNS Stack*

Description

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

Usage

```
NNS.stack(
  IVs.train,
  DV.train,
  IVs.test = NULL,
  type = NULL,
  obj.fn = expression(sum((predicted - actual)^2)),
  objective = "min",
  optimize.threshold = TRUE,
  dist = NULL,
  CV.size = NULL,
  balance = FALSE,
  ts.test = NULL,
  folds = 5,
  order = NULL,
  method = c(1, 2),
  stack = TRUE,
  dim.red.method = "cor",
  pred.int = NULL,
  status = TRUE,
  ncores = NULL,
  seed = 123L,
  .validation = NULL
)
```

Arguments

`IVs.train`	a vector, matrix or data frame of variables of numeric or factor data types.
`DV.train`	a numeric or factor vector with compatible dimensions to (`IVs.train`).

IVs.test — a vector, matrix or data frame of variables of numeric or factor data types with compatible dimensions to (IVs.train). If NULL, will use (IVs.train) as default. Columns are matched to (IVs.train) by name when the two share the same predictor names, and positionally when they share no names at all (as in cbind(test.x_1, test.x_2) against cbind(x_1, x_2)). A CSV UTF-8 byte-order mark in the first header (for example, Male becoming X...Male) is repaired before matching names. Other partial overlaps, including misspelled predictors, are ambiguous and return an error.

type — NULL (default) automatically classifies factor, character, logical and two-valued numeric responses. Set type = "CLASS" for nominal classification of any numeric labels, or type = "REG" to fit a numeric response as regression regardless of its number of distinct values. Forecasting calls use type = "REG".

obj.fn — expression; expression(sum((predicted - actual)^2)) (default) Sum of squared errors is the default objective function. Any expression() using the specific terms predicted and actual can be used.

objective — options: ("min", "max") "min" (default) Select whether to minimize or maximize the objective function obj.fn.

optimize.threshold — logical; TRUE (default) Optimizes the threshold for the second first-observed class in binary classification; multiclass decisions use the largest class probability. If FALSE, returns 0.5.

dist — options:(NULL, "NNS", "L1", "L2", "FACTOR") the method of distance calculation. dist = NULL is the default and selects the native blended NNS distance; dist = "NNS" is an explicit alias for the default. dist = "L2" selects Euclidean distance; dist = "L1" selects Manhattan distance; dist = "FACTOR" uses a frequency.

CV.size — numeric [0, 1]; NULL (default) Sets the holdout size for repeated holdout cross-validation (one random holdout per fold). With CV.size = NULL and folds > 1, (stratified, for classification) k-fold cross-validation is used; with CV.size = NULL and folds = 1, a single holdout of random size between 0.2 and 0.33 is used.

balance — logical; FALSE (default) Uses both up and down sampling to balance the classes. type="CLASS" required.

ts.test — integer; NULL (default) Sets the length of the test set for time-series data; typically 2*h parameter value from NNS.ARMA or double known periods to forecast.

folds — integer; folds = 5 (default) Select the number of cross-validation folds.

order — options: (integer, "max", NULL); NULL (default) Sets the order for NNS.reg, where (order = "max") is the k-nearest neighbors equivalent, which is suggested for mixed continuous and discrete (unordered, ordered) data.

method — numeric options: (1, 2); Select the NNS method to include in stack. (method = 1) selects NNS.reg; (method = 2) selects NNS.reg dimension reduction regression. Defaults to method = c(1, 2), which will reduce the dimension first, then find the optimal n.best.

stack	logical; TRUE (default) Uses dimension reduction to select actual predictor coordinates for the multivariate neighbour search; the dimension-reduction component separately uses its synthetic index. If FALSE, both components use their own full predictor designs.
dim.red.method	options: ("cor", "NNS.dep", "NNS.caus", "equal", "all") method for determining synthetic X* coefficients. (dim.red.method = "cor") (default) uses Spearman rank correlation for weights. (dim.red.method = "NNS.dep") uses NNS.dep for nonlinear dependence weights, while (dim.red.method = "NNS.caus") uses NNS.caus for directional association weights. (dim.red.method = "all") averages all methods for further feature engineering.
pred.int	numeric [0,1]; NULL (default) Returns the associated prediction intervals with each method.
status	logical; TRUE (default) Prints status update message in console.
ncores	integer; value specifying the number of cores to be used in the parallelized subroutine NNS.reg. If NULL (default), the number of cores to be used is equal to the number of cores of the machine - 1.
seed	Optional integer random seed used for reproducible resampling, fold construction, and stochastic fitting steps. If 'NULL', the current random-number-generator state is used.
.validation	Internal forecast-validation contract, normally NULL. A list with IVs.train, DV.train, IVs.test, and DV.test supplies the single chronological validation fold when ts.test is set, folds = 1, and CV.size = NULL. It requires exactly N-ts.test training rows and one through ts.test held-out rows; this allows scoring only observed future targets. Training prefixes and generated validation predictors must be constructed without later data. Used only for regression tuning; numeric two-level responses remain regression when this contract is supplied with type = NULL. The final model still fits all original training observations.

Details

Classification tunes and combines class-probability vectors from the NNS predictor cells, preserving nominal label identity. probabilities, reg.probabilities, and dim.red.probabilities return the final probability matrices. Class balancing resamples the training rows of each fold. The neighbour count and the dimension count are chosen by optimizing obj.fn on the pooled out-of-fold predictions; the default classification objective is accuracy. As in NNS 13.0, the two models' probability vectors are then blended with weights proportional to their squared objective scores (inverse squares when objective = "min"), so models with nearly equal scores are blended nearly equally, and the binary probability threshold is tuned on the out-of-fold predictions of that blend. OBJfn.reg and OBJfn.dim.red are the best scores of that search on the same predictions, so they are selection-optimized and overstate held-out performance; assess the final model on an independent test set. Custom objectives receive original numeric or logical response values, and factor/character codes as identified by class.levels. Relabelling is invariant for label-invariant objectives such as accuracy; a custom loss assigning different costs to classes can deliberately change the result.

Named numeric dimension-reduction coefficients are matched to encoded predictor names in both regression and classification. Factor columns use predictor__level names; unnamed vectors

follow encoded column order. Classification dimension reduction uses one signed index per class indicator. Its reported coefficients are a list of class-specific equations. The tuned dimension count is returned as `NNS.dim.red.count`, and `NNS.dim.red.threshold` gives the coefficient cutoff that keeps that count in every class index: one number when the class indices share it, as two-class indices always do, and otherwise one cutoff per class. The field `NNS.dim.red.variables` is the union of their active encoded columns. These equations describe the synthetic indices, not a complete saved prediction model: reconstruction also requires training-fitted scaling, the predictor cells, their class frequencies, and the selected smoothing controls. With `stack = TRUE`, the multivariate classification component retains the selected number of actual encoded coordinates ranked by maximum class-indicator dependence. `Prediction.set` contains classes attaining the requested estimated probability mass. Binary interval fields contain the range of this set for compatibility; multiclass interval fields are NULL. The sets do not assert calibrated coverage.

For regression, all dimension-reduction coefficients, normalizations, and candidate predictions used in validation are fitted to the corresponding training fold. The final coefficients and regressions are refitted to the full supplied training data. Method 1 retains the selected predictor coordinates, while method 2 evaluates their signed synthetic index; the components can therefore represent different regression geometry.

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; when all regression points (for classification, all cells) are selected, their number in the final fit. `NA` for a single predictor in regression, whose univariate NNS.reg fit has no `n.best` to tune (use `cbind(x, x)` to search one), and when `method` does not include 1. `"SSE.reg"` returns the SSE for the NNS.reg multivariate regression.
- `"OBJfn.reg"` returns the best `obj.fn` value of the NNS.reg regression's tuning on the out-of-fold predictions; it is selection-optimized, not an estimate of out-of-sample performance.
- `"NNS.dim.red.threshold"` returns the coefficient cutoff at which NNS.reg on `(IVs.train, DV.train)` with the same `dim.red.method` keeps the tuned number of predictors: the smallest kept coefficient magnitude, or 0 when every predictor is kept. Ties at the cutoff keep every tied predictor. For classification it is one number when every class index shares the cutoff, as two-class indices always do, and otherwise one cutoff per class, which NNS.reg accepts as `threshold`. `NA` when `method` does not include 2.
- `"NNS.dim.red.variables"` returns the names of the predictors carried by the final dimension reduction regression, ranked by absolute coefficient. With `(stack = TRUE)` these are also the only predictors used by the NNS.reg multivariate regression, so they are exactly the variables behind `"stack"`. `NULL` when `method` does not include 2.
- `"NNS.dim.red.coefficients"` returns the named final dimension reduction coefficients for every encoded predictor column (zero for predictors not selected). `NULL` when `method` does not include 2.
- `"OBJfn.dim.red"` returns the best `obj.fn` value of the NNS.reg dimension reduction regression's tuning on the out-of-fold predictions; it is selection-optimized, not an estimate of out-of-sample performance.
- `"probability.threshold"` returns the optimum probability threshold for classification, else 0.5 when set to `FALSE`.

- "reg" returns NNS.reg output.
- "reg.pred.int" returns the prediction intervals for the regression output.
- "dim.red" returns NNS.reg dimension reduction regression output.
- "dim.red.pred.int" returns the prediction intervals for the dimension reduction regression output.
- "stack" returns the output of the stacked model.
- "pred.int" returns the prediction intervals for the stacked model.
- "weights" returns the weights of "reg" and "dim.red" in "stack". For classification they are proportional to the squared "OBJfn.reg" and "OBJfn.dim.red" (inverse squares when objective = "min"); for regression the weight is selected by obj.fn on the out-of-fold predictions of the blend.

Note

- Incorporate any objective function from external packages (such as Metrics::mape) via NNS.stack(..., obj.fn = expression(Metrics::mape(actual, predicted)), objective = "min")
- Like a logistic regression, the (type = "CLASS") setting is not necessary for target variable of two classes e.g. [0, 1]. Numeric class values are returned on their original scale; factor and character responses use codes with the returned class.levels mapping.
- Missing data should be handled prior as well using na.omit or complete.cases on the full dataset.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

```
## Using 'iris' dataset with a stratified test set of ten rows per species
## (rows 141:150 alone are all virginica).
## Not run:
test.set <- c(41:50, 91:100, 141:150)
NNS.stack(iris[-test.set, 1:4], iris[-test.set, 5], IVs.test = iris[test.set, 1:4],
type = "CLASS", balance = TRUE)

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

## End(Not run)
```

NNS.TSD *NNS TSD Test*

Description

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

Usage

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

Arguments

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

Value

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

Note

The curves are compared on the merged sample points and, for TSD and continuous FSD, also at their closed-form extrema between those points. Differences smaller than 64 * .Machine$double.eps * S^d (S = largest absolute value in x and y, d = 1 for SSD and the mean condition, 2 for TSD; a ratio analogue for continuous FSD) are treated as ties, so that rounding noise does not decide dominance. Discrete FSD compares exact empirical CDF values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

x	a numeric vector.
y	a numeric vector.

Value

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

Note

The curves are compared on the merged sample points and, for TSD and continuous area order, also at their closed-form extrema between those points. Differences smaller than `64 * .Machine$double.eps * S^d` (S = largest absolute value in x and y, d = 1 for SSD and the mean condition, 2 for TSD; a ratio analogue for continuous area order) are treated as ties, so that rounding noise does not decide dominance. Discrete FSD compares exact empirical CDF values.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

## End(Not run)
```

NNS.VAR *NNS VAR*

Description

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

Usage

```
NNS.VAR(
  variables,
  h,
  tau = 1,
  dim.red.method = "cor",
  naive.weights = TRUE,
  obj.fn = expression({
     .denom <- NNS::Co.LPM(1, predicted, actual, target_x =
   mean(predicted), target_y = mean(actual)) + NNS::Co.UPM(1, predicted, actual,
    target_x = mean(predicted), target_y = mean(actual))
     if (.denom == 0) Inf else
    mean((predicted - actual)^2)/.denom
 }),
  objective = "min",
  status = TRUE,
  ncores = NULL,
  nowcast = FALSE,
  validation = c("pipeline", "observed")
)
```

Arguments

variables	a numeric matrix or data.frame of contemporaneous time-series to forecast.
h	integer; 1 (default) Number of periods to forecast. (h = 0) will return just the interpolated and extrapolated values.
tau	positive integer [> 0]; 1 (default) Number of lagged observations to consider for the time-series data. Vector for single lag for each respective variable or list for multiple lags per each variable.
dim.red.method	options: ("cor", "NNS.dep", "NNS.caus", "all") method for screening lagged regressors with NNS.reg before the multivariate NNS.stack fit. dim.red.method = "cor" (default) uses signed Spearman rank-correlation coefficients, screened by magnitude. "NNS.dep" uses nonlinear dependence weights, and "NNS.caus" uses directional association weights. "all" averages the Spearman, dependence, directional-association, and unit-weight coefficients. The coefficient cut-off is selected by scoring NNS.reg validation forecasts.

naive.weights	logical; TRUE (default) Equal weights applied to univariate and multivariate outputs in ensemble. FALSE will apply weights based on the number of relevant variables detected.
obj.fn	expression; expression(mean((predicted - actual)^2)) / (Sum of NNS Co-partial moments) (default) MSE / co-movements is the default objective function. Any expression(...) using the specific terms predicted and actual can be used.
objective	options: ("min", "max") "min" (default) Select whether to minimize or maximize the objective function obj.fn.
status	logical; TRUE (default) Prints status update message in console.
ncores	integer; value specifying the number of cores to be used in the parallelized subroutine NNS.ARMA.optim. If NULL (default), the number of cores to be used is equal to the number of cores of the machine - 1.
nowcast	logical; FALSE (default) internal call for frequency alignment in downstream nowcasting applications.
validation	"pipeline" (default) tunes with a chronological holdout whose predictors are built by completing and forecasting each series using only its prefix. Only observed holdout targets are scored. "observed" retains the former conditional-regression validation on realized covariates; it is not a deployment forecast score.

Value

Returns the following matrices of forecasted variables:

- "interpolated_and_extrapolated" Returns a data.frame of the original variables with their NA values replaced. When a variable's latest values are missing, every NA of that variable is estimated from its own observations by the NNS.stack fit on the time index used in NNS 13.0: NNS.reg with one nearest regression point, on the index duplicated into two predictors. The latest observation keeps its own regression point, and values after it (or before the first observation) extend the gradient at that boundary. When only earlier values are missing, they are interpolated linearly with NNS.reg(order = "max") on the index. 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: the lagged regressors whose dimension reduction coefficient meets the cross-validated threshold. These are the only regressors used in the "multivariate" estimate.
- "univariate" Returns the univariate NNS.ARMA forecasts.
- "multivariate" Returns the multi-variate forecasts: NNS.stack (method = 1, stack = FALSE) on the relevant variables, evaluated on the lag-matrix rows of the forecast period (built from the univariate forecasts).
- "validation" validation mode, origin and horizon used for tuning; pipeline mode excludes future covariates from prefix completion and prediction.
- "ensemble" Returns the ensemble of both "univariate" and "multivariate" forecasts.

Note

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

Author(s)

Fred Viole, OVVO Financial Systems

References

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

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

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

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

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

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

Examples

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

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

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

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

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

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

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

```
# Apply UPM.VaR and LPM.VaR for a band around the forecast path (2.5 percent per tail).
# The replicates resample the forecast path itself, so the band shows the path's
# dispersion, not a calibrated 95 percent prediction interval (see NNS.ARMA's pred.int).
# Tail percentage used in first argument per {LPM.VaR} and {UPM.VaR} functions
lower_CIs <- apply(replicates, 1, function(z) LPM.VaR(0.025, 0, z))
upper_CIs <- apply(replicates, 1, function(z) UPM.VaR(0.025, 0, z))

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


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

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

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


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

## End(Not run)
```

PM.matrix *Partial Moment Matrix*

Description

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

Usage

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

Arguments

LPM_degree	numeric; lower partial moment degree (0 = freq, 1 = area).
UPM_degree	numeric; upper partial moment degree (0 = freq, 1 = area).
target	numeric vector; thresholds for each column. There is no default: pass NULL to use colMeans(variable).
variable	numeric matrix or data.frame.

pop_adj	logical; TRUE applies the sample adjustment n / (n - 1) to the co-/divergent moments (with both degrees 1, target = NULL and norm = FALSE, the covariance matrix then equals cov()). It is ignored when either degree is 0.
norm	logical; default FALSE. If TRUE, each quadrant matrix is cell-wise normalized so their sum is 1 at each (i,j).

Details

Partial Moment Matrix

Value

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

Note

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

Examples

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

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

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

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

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

UPM *Upper Partial Moment*

Description

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

Usage

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

Arguments

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

Value

UPM of variable

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

UPM.ratio *Upper Partial Moment Ratio*

Description

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

Usage

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

Arguments

degree	numeric; degree = 0 gives frequency, degree = 1 gives area.
target	numeric vector; threshold(s). There is no default: pass NULL to use mean(variable).
variable	numeric vector or data-frame column to evaluate.

Value

Numeric vector of standardized upper partial moments. For `degree > 0`, a target at which every observation equals the target (0/0) returns the `degree = 0` ratio.

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

## End(Not run)
```

`UPM.VaR` *UPM VaR*

Description

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

Usage

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

Arguments

`percentile`	numeric [0, 1]; The percentile for right-tail VaR.
`degree`	integer; (`degree = 0`) gives the interpolated empirical quantile (`quantile(x, 1 - percentile, type = 7)`), which on tied data need not be an exact inverse of `UPM.ratio(0, ., x)`. Use it for probability statements. (`degree = 1`) gives an area-weighted (severity-weighted) threshold, the (1 - `percentile`)-expectile of `x` (Newey and Powell 1987): `percentile` of the area of deviations, not of the observations, lies above it, so it is not a quantile of `x`. Higher degrees weight large deviations more.
`x`	a numeric vector.

Value

Returns the type-7 empirical quantile (`degree = 0`), whose strict or inclusive sample tail fraction may differ from `percentile` because of interpolation or ties, or the threshold with `percentile` of the directional deviation area above it (`degree > 0`).

Author(s)

Fred Viole, OVVO Financial Systems

References

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

Examples

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

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

Index

Co.LPM, 3, *32*
Co.LPM_nD, 4
Co.UPM, 5, *32*
Co.UPM_nD, 6, *31*
complete.cases, *71*

D.LPM, 6
D.UPM, 7
data.frame, *13*, *49*, *79*
DPM_nD, 8
dy.d_, 11
dy.dx, 9

legend, *49*, *52*
list, *13*, *79*
LPM, 13
LPM.ratio, 14, *30*
LPM.VaR, 15, *20*, *23*, *26*, *63*

na.omit, *71*
NNS.ANOVA, 16
NNS.ARMA, 19, *21–23*, *26*, *61*, *68*, *74, 75*
NNS.ARMA.optim, *20*, 21, *60, 61*, *75*
NNS.boost, 24
NNS.caus, 27, *52*, *69*
NNS.CDF, 30
NNS.copula, 32
NNS.dep, *29*, 33, *45*, *48*, *52, 53*, *69*
NNS.diff, 35
NNS.distance, 37
NNS.FSD, 37, *64*
NNS.FSD.uni, 38
NNS.gravity, *17*, 40, *46*, *50*
NNS.MC, *20*, 41
NNS.meboot, *41, 42*, 42, *63*
NNS.mode, 46
NNS.moments, 47
NNS.norm, 48
NNS.part, *34*, 49
NNS.reg, *10–12*, *23–26*, *37*, 51, *67–71*, *74–76*
NNS.rescale, 56
NNS.SD.cluster, 57
NNS.SD.efficient.set, *58*, 59
NNS.seas, *19*, *28*, 60
NNS.SS, 62
NNS.SSD, *64*, 65
NNS.SSD.uni, 66
NNS.stack, *26*, *53*, 67, *74, 75*
NNS.TSD, *64*, 72
NNS.TSD.uni, 73
NNS.VAR, 74

PM.matrix, *32*, 77

UPM, 78
UPM.ratio, 79
UPM.VaR, *20*, *23*, *63*, 80