Package: knnwtsim (via r-universe)

August 20, 2024

Type Package

Title K Nearest Neighbor Forecasting with a Tailored Similarity Metric

Version 1.1.0.9000

Author Matthew Trupiano

Maintainer Matthew Trupiano <matthew.trupiano.professional@gmail.com>

Description Functions to implement K Nearest Neighbor forecasting using a weighted similarity metric tailored to the problem of forecasting univariate time series where recent observations, seasonal patterns, and exogenous predictors are all relevant in predicting future observations of the series in question. For more information on the formulation of this similarity metric please see Trupiano (2021) <arXiv:2112.06266>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://github.com/mtrupiano1/knnwtsim

BugReports https://github.com/mtrupiano1/knnwtsim/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.1.2

Imports stats

Depends R (>= 2.10)

Repository https://mtrupiano1.r-universe.dev

RemoteUrl https://github.com/mtrupiano1/knnwtsim

RemoteRef HEAD

RemoteSha 90ea40721d7f23cb736ffd6d0b16f423343abe46

Contents

`boston_911dispatch_weekly`
boston_911dispatch_weekly

Description

A dataset which contains the weekly count of 911 dispatches in the city of Boston, MA, USA, for three City of Boston public safety agencies: Boston Police Department, Boston Fire Department, and Boston Emergency Medical Services. In addition a number of holiday indicators, and dummy variables for months. Data is present for weeks between 2010-10-31 and 2014-04-20. Derived by Matthew Trupiano from a .csv file (911 Daily Dispatch Count By Agency (CSV)) hosted on https://data.boston.gov/.

Usage

```
boston_911dispatch_weekly
```

Format

A dataframe with 182 rows and 28 variables:

week first day of a given week (Sunday), date.

BPD summarized count of 911 dispatches from the Sunday in the week variable to the Saturday before the next week for the Boston Police Department.

EMS summarized count of 911 dispatches from the Sunday in the week variable to the Saturday before the next week for the Boston Emergency Medical Services.

BFD summarized count of 911 dispatches from the Sunday in the week variable to the Saturday before the next week for the Boston Fire Department.

new.years.ind 1 if YYYY-01-01 or YYYY-12-31 occur during the week else 0.

christmas.ind 1 if YYYY-12-24 or YYYY-12-25 occur during the week else 0.

thanksgiving.ind 1 if the holiday Thanksgiving occurs during the week else 0.

veterans.ind 1 if YYYY-11-11 occurs during the week else 0.

indigenous.ind 1 if the holiday Indigenous Peoples Day occurs during the week else 0.

labor.ind 1 if the holiday Labor Day occurs during the week else 0.

july4.ind 1 if YYYY-07-04 occurs during the week else 0.

juneteenth.ind 1 if YYYY-06-19 occurs during the week and the week st is >= '2020-06-01' else 0.

memorial.ind 1 if the holiday Memorial Day occurs during the week else 0.

patriots.ind 1 if the holiday Patriot's Day occurs during the week else 0.

st.patricks.ind 1 if YYYY-03-17 occurs during the week else 0.

presidents.ind 1 if the holiday President's Day occurs during the week else 0.

mlk.ind 1 if the holiday Martin Luther King Jr. Day occurs during the week else 0.

jan.ind 1 if week variable occurs in January else 0.

feb.ind 1 if week variable occurs in February else 0.

mar.ind 1 if week variable occurs in March else 0.

apr.ind 1 if week variable occurs in April else 0.

may.ind 1 if week variable occurs in May else 0.

jun.ind 1 if week variable occurs in June else 0.

jul.ind 1 if week variable occurs in July else 0.

aug.ind 1 if week variable occurs in August else 0.

sep.ind 1 if week variable occurs in September else 0.

oct.ind 1 if week variable occurs in October else 0.

nov.ind 1 if week variable occurs in November else 0.

dec.ind 1 if week variable occurs in December else 0.

Source

https://data.boston.gov/dataset/911-daily-dispatch-count-by-agency

---

`boston_fire_incidents_monthly`
boston_fire_incidents_monthly

Description

A dataset which contains the monthly count of fire incidents in the city of Boston, MA, USA, as well as one indicator for the period where a COVID-19 state of emergency was declared in Massachusetts. Data is present for starting from 2017-01-01 to 2021-07-01. Derived by Matthew Trupiano from a series of .csv files hosted on https://data.boston.gov/.

Usage

```
boston_fire_incidents_monthly
```

Format

A dataframe with 55 rows and 3 variables:

week first day of a given month, date.

incidents summarized count of fire incidents in the month which starts the date of the corresponding month variable.

covid.soe.ind 1 if the month contains days between '2020-03-10' and '2021-06-15' when a state of emergency for COVID-19 was declared in Massachusetts else 0.

Source

https://data.boston.gov/dataset/fire-incident-reporting

`boston_fire_incidents_weekly`
boston_fire_incidents_weekly

Description

A dataset which contains the weekly count of fire incidents in the city of Boston, MA, USA, as well as a number of holiday indicators and one indicator for the period where a COVID-19 state of emergency was declared in Massachusetts. Data is present for weeks between 2017-01-01 and 2021-07-25. Derived by Matthew Trupiano from a series of .csv files hosted on https://data.boston.gov/.

Usage

```
boston_fire_incidents_weekly
```

Format

A dataframe with 239 rows and 16 variables:

week first day of a given week (Sunday), date.

incidents summarized count of fire incidents from the Sunday in the week variable to the Saturday before the next week.

new.years.ind 1 if YYYY-01-01 or YYYY-12-31 occur during the week else 0.

christmas.ind 1 if YYYY-12-24 or YYYY-12-25 occur during the week else 0.

thanksgiving.ind 1 if the holiday Thanksgiving occurs during the week else 0.

veterans.ind 1 if YYYY-11-11 occurs during the week else 0.

indigenous.ind 1 if the holiday Indigenous Peoples Day occurs during the week else 0.

labor.ind 1 if the holiday Labor Day occurs during the week else 0.

july4.ind 1 if YYYY-07-04 occurs during the week else 0.

juneteenth.ind 1 if YYYY-06-19 occurs during the week and the week st is >= '2020-06-01' else 0.

memorial.ind 1 if the holiday Memorial Day occurs during the week else 0.

patriots.ind 1 if the holiday Patriot's Day occurs during the week else 0.

st.patricks.ind 1 if YYYY-03-17 occurs during the week else 0.

presidents.ind 1 if the holiday President's Day occurs during the week else 0.

mlk.ind 1 if the holiday Martin Luther King Jr. Day occurs during the week else 0.

covid.soe.ind 1 if the week contains days between '2020-03-10' and '2021-06-15' when a state of emergency for COVID-19 was declared in Massachusetts else 0.

Source

https://data.boston.gov/dataset/fire-incident-reporting

`knn.forecast` *KNN Forecast*

Description

Provide an n x n similarity matrix as input, all points both observed and those to be forecasted should be included. The `f.index.in` argument indicates which observations to identify neighbors for, and removes them from consideration as eligible neighbors. Once the matrix is subset down to only the columns in `f.index.in` and the rows excluding `f.index.in`, the `NNreg()` function is applied over the columns, returning for each column the mean of those points in `y.in` identified as neighbors based on the row index of the `k.in` most similar observations in the column. It is important that the index of the similarity matrix and `y.in` accurately reflect the time order of the observations.

Usage

```
knn.forecast(Sim.Mat.in, f.index.in, k.in, y.in)
```

Arguments

Sim.Mat.in	numeric and symmetric matrix of similarities (recommend use of S_w, see SwMatrixCalc()).
f.index.in	numeric vector indicating the indices of Sim.Mat.in and y.in which correspond to the time order of the points to be forecast.
k.in	integer value indicating the the number of nearest neighbors to be considered in forecasting, must be >= 1.
y.in	numeric vector of the response series to be forecast.

Value

numeric vector of the same length as f.index.in, of forecasted observations.

See Also

- NNreg() for the function used to perform knn regression on a single point.
- SwMatrixCalc() for the function to calculate a matrix with the recommended similarity measure.

Examples

```
Sim.Mat <- matrix(c(1, .5, .2, .5, 1, .7, .2, .7, 1),
  nrow = 3, ncol = 3, byrow = TRUE
)
y <- c(2, 1, 5)
f.index <- c(3)
k <- 2
knn.forecast(Sim.Mat.in = Sim.Mat, f.index.in = f.index, y.in = y, k.in = k)
```

---

knn.forecast.boot.intervals

KNN Forecast Bootstrap Prediction Intervals

Description

A function for forecasting using KNN regression with prediction intervals. The approach is based on the description of "Prediction intervals from bootstrapped residuals" from chapter 5.5 of Hyndman R, Athanasopoulos G (2021) https://otexts.com/fpp3/prediction-intervals.html#prediction-intervals-from-bootstrapped-residuals, modified as needed for use with KNN regression. The algorithm starts by calculating a pool of forecast errors to later sample from. If there are n points prior to the first observation indicated in f.index.in then there will be n - k.in errors generated by one-step ahead forecasts starting with the point of the response series at the index k.in + 1. The first k.in points cannot be estimated because a minimum of k.in eligible neighbors would

be needed. The optional burn.in argument can be used to increase the number of points from the start of the series that need to be available as neighbors before calculating errors for the pool. Next, B possible paths the series could take are simulated using the pool of errors. Each path is simulated by calling knn.forecast(), estimating the first point in f.index.in, adding a sampled forecast error, then adding this value to the end of the series. This process is then repeated for the next point in f.index.in until all have been estimated. The final output interval estimates are calculated for each point in f.index.in by taking the appropriate percentiles of the corresponding simulations of that point. The mean and medians are also calculated from these simulations. One important implication of this behavior is that the mean forecast output from this function can differ from the point forecast produced by knn.forecast() alone.

Usage

```
knn.forecast.boot.intervals(
  Sim.Mat.in,
  f.index.in,
  k.in,
  y.in,
  burn.in = NULL,
  B = 200,
  return.simulations = FALSE,
  level = 0.95
)
```

Arguments

Sim.Mat.in	numeric and symmetric matrix of similarities (recommend use of S_w, see SwMatrixCalc()).
f.index.in	numeric vector indicating the indices of Sim.Mat.in and y.in which correspond to the time order of the points to be forecast.
k.in	integer value indicating the the number of nearest neighbors to be considered in forecasting, must be >= 1.
y.in	numeric vector of the response series to be forecast.
burn.in	integer value which indicates how many points at the start of the series to set aside as eligible neighbors before calculating forecast errors to be re-sampled.
B	integer value representing the number of bootstrap replications, this will be the number of forecasts simulated and used to calculate outputs, must be >= 1.
return.simulations	logical value indicating whether to return all simulated forecasts.
level	numeric value over the range (0,1) indicating the confidence level for the prediction intervals.

Value

list of the following components:

lb numeric vector of the same length as f.index.in, with the estimated lower bound of the prediction interval.

ub numeric vector of the same length as `f.index.in`, with the estimated upper bound of the prediction interval.

mean numeric vector of the same length as `f.index.in`, with the mean of the `B` simulated paths for each forecasted point.

median numeric vector of the same length as `f.index.in`, with the median of the `B` simulated paths for each forecasted point.

simulated.paths numeric matrix where each of the `B` rows contains a simulated path for the points in `f.index.in`, only returned if `return.simulations = TRUE`.

See Also

- `knn.forecast()` for the function called to perform knn regression.
- `SwMatrixCalc()` for the function to calculate a matrix with the recommended similarity measure.
- Hyndman R, Athanasopoulos G (2021),"Forecasting: Principles and Practice, 3rd ed", Chapter 5.5, https://otexts.com/fpp3/prediction-intervals.html#prediction-intervals-from-bootstrapped-re For background on the algorithm this function is based on.

Examples

```
data("simulation_master_list")
series.index <- 15
ex.series <- simulation_master_list[[series.index]]$series.lin.coef.chng.x

# Weights pre tuned by random search. In alpha, beta, gamma order
pre.tuned.wts <- c(0.2148058, 0.2899638, 0.4952303)
pre.tuned.k <- 5

df <- data.frame(ex.series)
# Generate vector of time orders
df$t <- c(1:nrow(df))

# Generate vector of periods
nperiods <- simulation_master_list[[series.index]]$seasonal.periods
df$p <- rep(1:nperiods, length.out = nrow(df))

# Pull corresponding exogenous predictor(s)
X <- as.matrix(simulation_master_list[[series.index]]$x.chng)


# Calculate the weighted similarity matrix using Sw
Sw.ex <- SwMatrixCalc(
  t.in = df$t,
  p.in = df$p, nPeriods.in = nperiods,
  X.in = X,
  weights = pre.tuned.wts
)

n <- length(ex.series)
# Index we want to forecast
```

```
f.index <- c((n - 5 + 1):length(ex.series))

interval.forecast <- knn.forecast.boot.intervals(
  Sim.Mat.in = Sw.ex,
  f.index.in = f.index,
  y.in = ex.series,
  k.in = pre.tuned.k
)
```

`knn.forecast.randomsearch.tuning`

Tune `knn.forecast()` *Hyperparameters with Random Search*

Description

A simplistic automated hyperparameter tuning function which randomly generates a grid of hyperparameter sets used to build corresponding `S_w` similarity matrices which are used in `knn.forecast()` test against the last `test.h` points of `y.in` after any `val.holdout.len` points are removed from the end of `y.in`. The best performing set of parameters based on MAPE over over the forecast horizon of `test.h` points are returned as part of a list alongside the 'optimum' weighted similarity matrix `Sw.opt`, the `Grid` of tested sets, and the MAPE results. MAPE is the average of absolute percent errors for each point calculated as: `abs((test.actuals - test.forecast.i) / test.actuals) * 100`. Where `test.forecast.i` and `test.actuals` are both numeric vectors.

Usage

```
knn.forecast.randomsearch.tuning(
  grid.len = 100,
  St.in,
  Sp.in,
  Sx.in,
  y.in,
  test.h = 1,
  max.k = NULL,
  val.holdout.len = 0,
  min.k = 1
)
```

Arguments

`grid.len`	integer value representing the number of hyperparameter sets to generate and test, must be >= 1.
`St.in`	numeric and symmetric matrix of similarities, can be generated with `StMatrixCalc()`.
`Sp.in`	numeric and symmetric matrix of similarities, can be generated with `SpMatrixCalc()`.
`Sx.in`	numeric and symmetric matrix of similarities, can be generated with `SxMatrixCalc()`.
`y.in`	numeric vector of the response series to be forecast.

`test.h` integer value representing the number of points in the test forecast horizon, must be >= 1.

`max.k` integer value representing the maximum value of k, `knn.forecast()` should use, will be set to `min(floor((length(y.in)) * .4), length(y.in) - val.holdout.len - test.h)` if NULL or NA is passed. Note this NA behavior differs from knnwtsim version 0.1.0. If a numeric value is passed it must be >= 1.

`val.holdout.len` integer value representing the number of observations at the end of the series to be removed in testing forecast if desired to leave a validation set after tuning, must be >= 0.

`min.k` integer value representing the minimum value of k, `knn.forecast()` should use, must be >= 1.

Value

list of the following components:

weight.opt numeric vector of the 3 weights to generate `Sw.opt` in alpha, beta, gamma order which achieved the best performance in terms of MAPE.

k.opt integer value of neighbors used in `knn.forecast()` which achieved the best performance in terms of MAPE.

Sw.opt numeric matrix of similarities calculated using `S_w`, with the best performing set of hyperparameters.

Test.MAPE numeric value of the MAPE result for the optimum hyperparamter set achieved on the test points.

MAPE.all numeric vector of MAPE results, each observation corresponds to the row in `Grid` of the same index.

Grid dataframe of all hyperparameter sets tested in the tuning.

See Also

- Trupiano (2021) arXiv:2112.06266 for information on the formulation of `S_w`.
- `StMatrixCalc()` for information on the calculation of `S_t`.
- `SpMatrixCalc()` for information on the calculation of `S_p`.
- `SxMatrixCalc()` for information on the calculation of `S_x`.
- `knn.forecast()` for the function called to perform knn regression.

Examples

```
data("simulation_master_list")
series.index <- 15
ex.series <- simulation_master_list[[series.index]]$series.lin.coef.chng.x

df <- data.frame(ex.series)
# Generate vector of time orders
df$t <- c(1:nrow(df))
# Generate vector of periods
```

```
nperiods <- simulation_master_list[[series.index]]$seasonal.periods
df$p <- rep(1:nperiods, length.out = nrow(df))
# Pull corresponding exogenous predictor(s)
X <- as.matrix(simulation_master_list[[series.index]]$x.chng)

St.ex <- StMatrixCalc(df$t)
Sp.ex <- SpMatrixCalc(df$p, nPeriods = nperiods)
Sx.ex <- SxMatrixCalc(X)

tuning.test <- knn.forecast.randomsearch.tuning(
  grid.len = 10,
  y.in = ex.series,
  St.in = St.ex,
  Sp.in = Sp.ex,
  Sx.in = Sx.ex,
  test.h = 3,
  max.k = 10,
  val.holdout.len = 3
)
```

`NNreg` *Estimate a Single Point with K Nearest Neighbors Regression*

Description

Finds the index of the nearest neighbors for a single point given that point's vector of similarities to all observations eligible to be considered as neighbors. The `k.in2` neighbors are identified by their index in the similarity vector, and this index is used to identify the neighbor points in `y.in2`. The function then returns the mean of the values in `y.in2` identified as neighbors. It is suggested to call this function through `knn.forecast()` for all points to be forecasted simultaneously.

Usage

```
NNreg(v, k.in2, y.in2)
```

Arguments

`v`	numeric vector of similarities used to identify nearest neighbors.
`k.in2`	integer value indicating the the number of nearest neighbors to be considered.
`y.in2`	numeric vector of the response series to be forecast.

Value

numeric value of the mean of the `k.in2` nearest neighbors in `y.in2`.

See Also

`knn.forecast()` the recommended user facing function to perform knn regression for forecasting with `NNreg()`.

Examples

```
Sim.Mat <- matrix(c(1, .5, .2, .5, 1, .7, .2, .7, 1),
  nrow = 3, ncol = 3, byrow = TRUE
)
Sim.Mat.col <- Sim.Mat[-(3), 3]
y <- c(2, 1, 5)
k <- 2
NNreg(v = Sim.Mat.col, k.in2 = 2, y.in2 = y)
```

SeasonalAbsDissimilarity

Seasonal Absolute Dissimilarity

Description

Calculate seasonal dissimilarity measure between the respective seasonal period two points, given the number of periods in one full seasonal cycle.

Usage

```
SeasonalAbsDissimilarity(p1, p2, nPeriods)
```

Arguments

p1	numeric value representing a seasonal period.
p2	numeric value representing a seasonal period.
nPeriods	numeric value representing the maximum value p1 or p2 can take on.

Value

numeric value of the seasonal dissimilarity between p1 and p2.

See Also

Trupiano (2021) arXiv:2112.06266 for information on the formulation of this seasonal dissimilarity measure.

Examples

```
SeasonalAbsDissimilarity(1, 4, 4)
```

simulation_master_list

simulation_master_list

Description

A list of 20 lists. Each of the 20 lists contains 31 items including 4 simulated time series. Each series contains an ARIMA component, a periodic component simulated using trig functions, a component determined by a functional relationship to exogenous predictors which we will call the f(x) component, a constant, and finally additional noise generated from either a Gaussian distribution with mean = 0, or Poisson distribution. The 4 series within a given sublist only differ based on the f(x) component of the series. One series, `series.mvnormx`, uses a matrix `X` generated by `MASS::mvrnorm()` with corresponding coefficients for the f(x) component. All other series use piece-wise functional relationships for the f(x) component of the series.

Usage

```
simulation_master_list
```

Format

A list containing 20 sublists each with 31 items:

series.len the number of observations in the simulated time series

random.seed the random seed used in `set.seed()` for all random components in the sublist.

arima.p The AR order argument for `stats::arima.sim()`.

arima.d The differencing order argument for `stats::arima.sim()`.

arima.q The MA order argument for `stats::arima.sim()`.

ar.coefficients Coefficients for the AR process in `stats::arima.sim()`,`NULL` if `arima.p=0`.

ma.coefficients Coefficients for the MA process in `stats::arima.sim()`,`NULL` if `arima.q=0`.

seasonal.periods The number of periods in a full cycle for the periodic component of the series.

sin.coef Coefficient on the sin term of the periodic component of the series.

cos.coef Coefficient on the cos term of the periodic component of the series.

X.cols Number of predictors used to generate the f(x) component of `series.mvnormx`.

X.mu The mean vector used in `MASS::mvrnorm()` to generate `X` for the f(x) component of `series.mvnormx`.

X.Sigma The covariance matrix generated by `clusterGeneration::rcorrmatrix()` used in `MASS::mvrnorm()` to generate `X` for the f(x) component of `series.mvnormx`.

X The matrix of `X.cols` predictors generated by `MASS::mvrnorm()` used to generate the f(x) component of `series.mvnormx`.

x.coef The vector of `X.cols` coefficients corresponding to the predictors of `X` used to generate the f(x) component of `series.mvnormx`.

x.chng.mu The mean value used in `stats::rnorm()` used to generate `x.chng`.

x.chng.sd The standard deviation value used in `stats::rnorm()` used to generate `x.chng`.

x.chng.coef1 A coefficient for `x.chng` used in all piece-wise functional relationship, f(x), components, as the `coef` argument to `lin.to.sqrt()` and `quad.to.cubic` and the `coef1` argument to `lin.coef.change` .

x.chng.coef2 A coefficient for `x.chng` used in the piece-wise functional relationship, f(x), component of `series.lin.coef.chng.x`, as the `coef2` argument to `lin.coef.change` .

x.chng.break.point A value used in two piece-wise functional relationship, f(x), components, as the `break.point` argument to `quad.to.cubic` and `lin.coef.change` .

x.chng.break.point.sqrt The `max()` of `x.chng.break.point` and some value > 0. Used in the piece-wise functional relationship, f(x), component of `series.lin.to.sqrt.x` , as the `break.point` argument to `lin.to.sqrt`.

x.chng A vector of observations of a single predictor used to generate the f(x) component of all series other than `series.mvnormx`.

type.noise The family of probability distributions to to generate the additional noise component.

poisson.rate The `lambda` argument of `stats::rpois()` used to generate additional noise, only actually used if `type.noise = 'poisson'`.

norma.sd The `sd` argument of `stats::rnorm()` used to generate additional noise, only actually used if `type.noise = 'normal'`.

constant A numeric value which is the constant component of the series.

series.mvnormx A simulated time series generated from the sum of ARIMA, Periodic, f(x), noise, and constant components. In this case f(x) represents linear relationships to the columns of the matrix `X`.

series.lin.to.sqrt.x A simulated time series generated from the sum of ARIMA, Periodic, f(x), noise, and constant components. In this case f(x) represents a linear relationship to a single predictor `x.chng` which changes to a `sqrt(x.chng)` relationship when `x.chng > x.chng.break.point.sqrt`.

series.lin.coef.chng.x A simulated time series generated from the sum of ARIMA, Periodic, f(x), noise, and constant components. In this case f(x) represents a linear relationship to a single predictor `x.chng` which changes coefficient when `x.chng > x.chng.break.point`.

series.quad.to.cubic.x A simulated time series generated from the sum of ARIMA, Periodic, f(x), noise, and constant components. In this case f(x) represents a quadratic relationship to a single predictor `x.chng` which changes to a cubic relationship when `x.chng > x.chng.break.point`, in addition a coefficient changes sign at `x.chng.break.point`.

Details

Below we have the functional relationships used for the piece-wise series:

```
lin.to.sqrt <- function(x, break.point, coef){ if (x < break.point) { out <- coef * x } else { out <- sqrt(x) } return(out) }
```

```
quad.to.cubic <- function(x, break.point, coef){ if (x < break.point) { out <- coef * (x ** 2) } else { out <- -coef * (x ** 3) } return(out) }
```

```
 lin.coef.change <- function(x, break.point, coef1, coef2){ if (x < break.point) { out <- coef1 * x } else { out <- coef2 * x } return(out) }
```

Source

https://github.com/mtrupiano1/knnwtsim/blob/main/data-raw/simulation_master_list.R

`SpMatrixCalc` *Calculate Seasonal Similarity Matrix*

Description

Generates and returns an n x n matrix by calculating the seasonal dissimilarity for each possible pair of points in a vector of seasonal periods, then converts dissimilarity matrix to a similarity matrix using 1 / (D_p + 1).

Usage

```
SpMatrixCalc(v, nPeriods)
```

Arguments

`v`	positive numeric vector with the seasonal periods corresponding to each point in the response series.
`nPeriods`	positive numeric value representing the maximum value `v` can take on.

Value

numeric matrix of seasonal similarities for the vector `v`.

See Also

- Trupiano (2021) arXiv:2112.06266 for information on the formulation of this seasonal similarity measure.
- `SeasonalAbsDissimilarity()` for the function used to calculate seasonal dissimilarity.

Examples

```
SpMatrixCalc(c(1, 2, 4), 4)
```

StMatrixCalc *Calculate Temporal Similarity Matrix*

Description

Generates and returns an n x n matrix by calculating the absolute difference for each possible pair of points in a vector of the time orders of each point in a series, then converts dissimilarity matrix to a similarity matrix using 1 / (D_t + 1).

Usage

```
StMatrixCalc(v)
```

Arguments

v	numeric vector with the time order corresponding to each point in the response series.

Value

numeric matrix of temporal similarities for the vector v.

See Also

`TempAbsDissimilarity()` for the function used to calculate absolute differences.

Examples

```
StMatrixCalc(c(1, 2, 3))
```

SwMatrixCalc *Calculate Weighted Similarity Matrix*

Description

A wrapper function which calls each of `StMatrixCalc(v = t.in)`, `SpMatrixCalc(v = p.in, nPeriods = nPeriods.in)`, and `SxMatrixCalc(A = X.in, XdistMetric = XdistMetric.in)` to generate the three matrices of using the component measures of `S_w`. Then generates the final weighted similarity matrix as the sum of each component matrix multiplied by its corresponding `weights`. The first value in `weights` will be multiplied by `S_t`, the second `S_p`, and the third `S_x`.

Usage

```
SwMatrixCalc(
  t.in,
  p.in,
  nPeriods.in,
  X.in,
  XdistMetric.in = "euclidean",
  weights = c(1/3, 1/3, 1/3)
)
```

Arguments

t.in	numeric vector of time orders for points in the response series.
p.in	numeric vector of period within a seasonal cycle (ex. 1 for January points in monthly data).
nPeriods.in	numeric scalar indicating the maximum value p.in could take on (ex. 12 for monthly data).
X.in	numeric vector or matrix of exogenous predictors, where the rows correspond to points in the response series.
XdistMetric.in	character describing the method stats::dist() should use. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary", or "minkowski".
weights	numeric vector where first value represents weight for S_t, second value the weight for S_p, and the third value the weight for S_x.

Value

numeric matrix of similarities which is calculated using S_w.

See Also

- Trupiano (2021) arXiv:2112.06266 for information on the formulation of S_w.
- StMatrixCalc() for information on the calculation of S_t.
- SpMatrixCalc() for information on the calculation of S_p.
- SxMatrixCalc() for information on the calculation of S_x.

Examples

```
t <- c(1, 2, 3)
p <- c(1, 2, 1)
X <- matrix(c(1, 1, 1, 2, 2, 2, 3, 3, 3), nrow = 3, ncol = 3, byrow = TRUE)
SwMatrixCalc(
  t.in = t,
  p.in = p, nPeriods.in = 2,
  X.in = X,
  weights = c(1 / 4, 1 / 4, 1 / 2)
)
```

SxMatrixCalc *Calculate Similarity Matrix for Exogenous Predictors*

Description

Largely a wrapper function for the `stats::dist()` function. First calculates n x n distance matrix using specified method for an input matrix or vector using `stats::dist()`. Then converts the distance matrix to similarity matrix using 1 / (D_x + 1).

Usage

```
SxMatrixCalc(A, XdistMetric = "euclidean")
```

Arguments

`A`	numeric matrix or numeric vector where the columns represents exogenous predictor variables and the rows correspond to the points in the response series.
`XdistMetric`	character describing the method `stats::dist()` should use. This must be one of `"euclidean"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"`, or `"minkowski"`.

Value

numeric matrix of distances for `A`.

Examples

```
X <- matrix(c(1, 1, 1, 2, 2, 2, 3, 3, 3), nrow = 3, ncol = 3, byrow = TRUE)
SxMatrixCalc(X)
```

TempAbsDissimilarity *Temporal Absolute Dissimilarity*

Description

Simply takes the absolute difference between two points, meaning points close in time will have smaller dissimilarity. This is equivalent to Euclidean Distance.

Usage

```
TempAbsDissimilarity(p1, p2)
```

Arguments

`p1`	numeric value representing the time order of the point in the response series.
`p2`	numeric value representing the time order of the point in the response series.

Value

numeric value of the absolute difference between p1 and p2.

Examples

```
TempAbsDissimilarity(1, 3)
```

Index