Package ‘kdml’

August 27, 2024

Title Kernel Distance Metric Learning for Mixed-Type Data

Version 1.0.0

Maintainer John R. J. Thompson <john.thompson@ubc.ca>

Description Distance metrics for mixed-type data consisting of continuous, nominal, and ordinal variables. This methodology uses additive and product kernels to calculate similarity functions and metrics, and selects variables relevant to the underlying distance through bandwidth selection via maximum similarity cross-validation. These methods can be used in any distance-based algorithm, such as distance-based clustering. For further details, we refer the reader to Ghashti and Thompson (2024) <<doi:10.48550/arXiv.2306.01890>> for dkps() methodology, and Ghashti (2024) <doi:10.14288/1.0443975> for dkss() methodology.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.5.0), np

Imports MASS, markdown

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Jesse S. Ghashti [aut],
John R. J. Thompson [aut, cre]

Repository CRAN

Date/Publication 2024-08-27 11:30:05 UTC

Contents

confactord 2
dkps 4
dkss 7
kdml 10
mscv.dkps 11
mscv.dkss 14

Index **17**

confactord *Mixed-type Data Generation with True Membership Labels*

Description

This function generates a mixed-type data frame with a combination of continuous (`numeric`), nominal (`factor`), and ordinal (`ordered`) variables with prespecified cluster overlap for each variable type. `confactord` allows the user to specify the number of each variable type, the amount of variables per variable type that have cluster overlap, the amount of cluster overlap for each variable type, the number of levels for the nominal and ordinal variables, and proportion of observations per class membership. Within and across-type variables are generated independently from one another. Currently, only two classes are may be generated.

Usage

```
confactord(n = 200,
            popProb = c(0.5,0.5),
            numMixVar = c(1,1,1),
            numMixVarOl = c(1,1,1),
            olVarType = c(0.1,0.1,0.1),
            catLevels = c(2,4))
```

Arguments

`n`	integer number of observations to be generated. Defaults to `n = 200`
`popProb`	numeric vector of length two specifying the proportion of observations allocated to each class membership, which must sum to one. Defaults to `popProb = c(0.5, 0.5)`.
`numMixVar`	numeric vector of integers of length three specifying (in order) the total number of continuous (numeric), nominal (factor), and ordinal (ordered) variables to be generated. If a specific variable type is not required, set the appropriate vector indice to zero. Defaults to `numMixVar = c(1,1,1)`.
`numMixVarOl`	numeric vector of integers of length three specifying (in order) the total number of continuous (numeric), nominal (factor), and ordinal (ordered) variables that will have class membership overlap. If all variables are to be well-separated by class membership, set all indices to zero. No indice of this vector may be greater than the corresponding indice in `numMixVar`. Defaults to `numMixVarOl = c(1,1,1)`.
`olVarType`	numeric vector of length three specifying (in order) the percentage of class membership overlap to be applied to the continuous (numeric), nominal (factor), and ordinal (ordered) No argument required if `numMixVarOl = c(0,0,0)`. Permissible class membership overlap per variable type is between 0.01 and 0.99. Defaults to ten percent overlap per variable type, `olVarType = c(0.1,0.1,0.1)`.
`catLevels`	numeric vector of length two specifying (in order) the number of levels (integer values) for each of the nominal (factor) and ordinal (ordered) variable types. Defaults to `catLevels = c(2,4)`.

Details

Continuous variables are generated independently from normal distributions, with means determined by true class membership. If overlap is specified, additional variance is introduced to simulate cluster overlap. Nominal variables are generated using Dirichlet distributions representing different population proportions. Ordinal variables are initially simulated as continuous variables and then discretized into ordered categories based on quantile distributions, similar to a latent class model where ordinal categories are inferred based on underlying continuous distributions and adjusted for cluster overlap parameters.

Value

`confactord` returns a `list` object, with the following components:

`data`	a `data.frame` of mixed variable types based on user- specified parameters
`class`	a numeric vector of integers specifying the true class memberships for the returned `data` data frame

Author(s)

John R. J. Thompson <john.thompson@ubc.ca>, Jesse S. Ghashti <jesse.ghashti@ubc.ca>

See Also

`mscv.dkss`, `mscv.dkps`, `dkss`, `dkps`

Examples

```
# EXAMPLE1: Default implementation generates the following
# 200 observations split into two clusters of equal size (100 observations each)
# Three variables-- one of each numeric, factor, and ordered
# Each variable has ten percent cluster overlap
# Nominal variable is binary
# Ordinal variable has four levels

df1 <- confactord()


# EXAMPLE2:
# 500 observations; 100 observations in cluster one and 400 in cluster two
# Three continuous variables, two nominal, one ordinal
# Only one continuous variable has cluster overlap
# All nominal and ordinal variables have cluster overlap
# Cluster overlap for continuous variable is twenty percent
# Cluster overlap for nominal variables are thirty percent
# Cluster overlap for ordinal variable is fourty percent
# Nominal variable has three levels, while ordinal has 5

df2 <- confactord(n = 500,
                    popProb = c(0.2,0.8),
                    numMixVar = c(3,2,1),
                    numMixVarOl = c(1,2,1),
```

```
olVarType = c(0.2,0.3,0.4),
catLevels = c(3,5))
```

---

dkps *Distance using Kernel Product Similarity (DKPS) for Mixed-type Data*

Description

This function calculates the pairwise distances between mixed-type observations consisting of continuous (numeric), nominal (factor), and ordinal (ordered) variables using the kernel product similarity described in Ghashti and Thompson (2024). This kernel metric learning methodology calculates a kernel product similarity function, with a variety of options for kernel functions associated with each variable type and returns a distance matrix that can be used in any distance-based algorithm.

Usage

```
dkps(df, bw = "mscv", cFUN = "c_gaussian", uFUN = "u_aitken",
      oFUN = "o_wangvanryzin", stan = TRUE, verbose = FALSE)
```

Arguments

df	a p-variate data frame for which the pairwise distances between observations will be calculated. The data types may be continuous (numeric), nominal (factor), and ordinal (ordered), or any combination thereof. Columns of df should be of appropriate variable type prior to running the function.
bw	numeric bandwidth vector of length p, with each element i corresponding to the bandwidth for column i in df. Alternatively, one of two character strings may be inputted for bandwidth selection methods. mscv specifies maximum-similarity cross-validation, and np specifies likelihood-cross validation which is calculated using npudensbw in package np. Defaults to mscv. See details.
cFUN	character value specifying the continuous kernel function. Options include c_gaussian, c_epanechnikov, c_uniform, c_triangle, c_biweight, c_triweight, c_tricube, c_cosine, c_logistic, c_sigmoid, and c_silverman. Note that if using np for bw selection above, continuous kernel types are restricted to either c_gaussian, c_epanechnikov, or c_uniform. Defaults to c_gaussian. See details.
uFUN	character value specifying the nominal kernel function for unordered factors. Options include u_aitken and u_aitchisonaitken. Defaults to u_aitken. See details.
oFUN	character value specifying the ordinal kernel function for ordered factors. Options include o_aitken, o_aitchisonaitken, o_habbema, o_wangvanryzin, and o_liracine. Note that if using np for bw selection above, ordinal kernel types are restricted to either o_wangvanryzin or o_liracine. Defaults to o_wangvanryzin. See details.

`stan`	a logical value which specifies whether to scale the resulting distance matrix between 0 and 1 using min-max normalization. If set to `FALSE`, distances are unscaled. Defaults to `TRUE`.
`verbose`	a logical value which specifies whether to print procedural steps to the console. If set to `FALSE`, no output is printed to the console. Defaults to `FALSE`.

Details

`dkps` implements the distance using kernel product similarity (DKPS) as described by Ghashti and Thompson (2024). This approach uses product kernels for continuous variables, and summation kernels for nominal and ordinal data, which are then summed over all variable types to return the pairwise distance between mixed-type data.

There are several kernels to select from. The continuous kernel functions may be found in Cameron and Trivedi (2005), Härdle et al. (2004) or Silverman (1986). Nominal kernels use a variation on Aitchison and Aitken's (1976) kernel, while ordinal kernels use a variation of the Wang and van Ryzin (1981) kernel. Both nominal and ordinal kernel functions can be found in Li and Racine (2007), Li and Racine (2003), Ouyan et al. (2006), and Titterington and Bowman (1985).

Each kernel requires a bandwidth specification, which can either be a user defined numeric vector of length p from alternative methodologies for bandwidth selection, or through two bandwidth selection methods can be specified. The `mscv` bandwidth selection is based on maximum similarity cross-validation by Ghashti and Thompson (2024), invoked by the function `mscv.dkss`. The `np` bandwidth selection follows the maximum likelihood cross-validation method described by Li and Racine (2007) and Li and Racine (2003) for kernel density estimation of mixed-type data.

Data contained in the data frame `df` may constitute any combinations of continuous, nominal, or ordinal data, which is to be specified in the data frame `df` using `factor` for nominal data, and `ordered` for ordinal data. Data types can be in any order and will be detected automatically. User-inputted vectors of bandwidths `bw` must be specified in the same order as the variables in the data frame `df`, as to ensure they sorted accordingly by the routine.

Value

`dkps` returns a `list` object, with the following components:

`distances`	an $n \times n$ numeric matrix containing pairwise distances between observations
`bandwidths`	a p-variate vector of bandwidth values returned based on the `bw` bandwidth specification method, sorted by variable type

Author(s)

John R. J. Thompson <john.thompson@ubc.ca>, Jesse S. Ghashti <jesse.ghashti@ubc.ca>

References

Aitchison, J. and C.G.G. Aitken (1976), "Multivariate binary discrimination by the kernel method," Biometrika, 63, 413-420.

Cameron, A. and P. Trivedi (2005), "Microeconometrics: Methods and Applications", Cambridge University Press.

Ghashti, J.S. and J.R.J Thompson (2024), "Mixed-type Distance Shrinkage and Selection for Clustering via Kernel Metric Learning." Journal of Classification, Accepted.

Härdle, W., and M. Müller and S. Sperlich and A. Werwatz (2004), *Nonparametric and Semiparametric Models*, (Vol. 1). Berlin: Springer.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Li, Q. and J.S. Racine (2003), "Nonparametric estimation of distributions with categorical and continuous data," Journal of Multivariate Analysis, 86, 266-292.

Ouyang, D. and Q. Li and J.S. Racine (2006), "Cross-validation and the estimation of probability distributions with categorical data," Journal of Nonparametric Statistics, 18, 69-100.

Silverman, B.W. (1986), *Density Estimation,* London: Chapman and Hall.

Titterington, D.M. and A.W. Bowman (1985), "A comparative study of smoothing procedures for ordered categorical data", Journal of Statistical Computation and Simulation, 21(3-4), 291-312.

Wang, M.C. and J. van Ryzin (1981), "A class of smooth estimators for discrete distributions," Biometrika, 68, 301-309.

See Also

mscv.dkps, dkss, mscv.dkss

Examples

```
# example data frame with mixed numeric, nominal, and ordinal data.
levels = c("Low", "Medium", "High")
df <- data.frame(
  x1 = runif(100, 0, 100),
  x2 = factor(sample(c("A", "B", "C"), 100, TRUE)),
  x3 = factor(sample(c("A", "B", "C"), 100, TRUE)),
  x4 = rnorm(100, 10, 3),
  x5 = ordered(sample(c("Low", "Medium", "High"), 100, TRUE), levels = levels),
  x6 = ordered(sample(c("Low", "Medium", "High"), 100, TRUE), levels = levels))

# minimal implementation requires just the data frame, and will automatically be
# defaulted to the mscv bandwidth specification technique and default kernel
# function
d1 <- dkps(df = df)
# d$bandwidths to see the mscv obtained bandwidths
# d$distances to see the distance matrix


# try using the np package, which has few continuous and ordinal kernels to
# choose from. Recommended using default kernel functions
d2 <- dkps(df = df, bw = "np")


# precomputed bandwidth example
# note that continuous variables requires bandwidths > 0
# ordinal variables requires bandwidths in [0,1]
# for nominal variables, u_aitken requires bandwidths in [0,1]
```

```
# and u_aitchisonaitken in [0,(c-1)/c]
# where c is the number of unique values in the i-th column of df.
# any bandwidths outside this range will result in a warning message
bw_vec <- c(1.0, 0.5, 0.5, 5.0, 0.3, 0.3)
d3 <- dkps(df = df, bw = bw_vec)


# user-specific kernel functions example
d5 <- dkps(df = df, bw = "mscv", cFUN = "c_epanechnikov", uFUN = "u_aitken",
      oFUN = "o_habbema")
```

`dkss` *Distance using Kernel Summation Similarity (DKSS) for Mixed-type Data*

Description

This function calculates the pairwise distances between mixed-type observations consisting of continuous (`numeric`), nominal (`factor`), and ordinal (`ordered`) variables using the method described in Ghashti (2024). This kernel metric learning methodology calculates a kernel sum similarity function, with a variety of options for kernel functions associated with each variable type and returns a distance matrix that can be used in any distance- based algorithm.

Usage

```
dkss(df, bw = "mscv", cFUN = "c_gaussian", uFUN = "u_aitken",
    oFUN = "o_wangvanryzin", stan = TRUE, verbose = FALSE)
```

Arguments

`df`	a p-variate data frame for which the pairwise distances between observations will be calculated. The data types may be continuous (`numeric`), nominal (`factor`), and ordinal (`ordered`), or any combination thereof. Columns of `df` should be of appropriate variable type prior to running the function.
`bw`	numeric bandwidth vector of length p, with each element i corresponding to the bandwidth for column i in `df`. Alternatively, one of two character strings may be inputted for bandwidth selection methods. `mscv` specifies maximum-similarity cross-validation, and `np` specifies likelihood-cross validation which is calculated using `npudensbw` in package `np`. Defaults to `mscv`. See details.
`cFUN`	character value specifying the continuous kernel function. Options include `c_gaussian`, `c_epanechnikov`, `c_uniform`, `c_triangle`, `c_biweight`, `c_triweight`, `c_tricube`, `c_cosine`, `c_logistic`, `c_sigmoid`, and `c_silverman`. Note that if using `np` for `bw` selection above, continuous kernel types are restricted to either `c_gaussian`, `c_epanechnikov`, or `c_uniform`. Defaults to `c_gaussian`. See details.
`uFUN`	character value specifying the nominal kernel function for unordered factors. Options include `u_aitken` and `u_aitchisonaitken`. Defaults to `u_aitken`. See details.

oFUN	character value specifying the ordinal kernel function for ordered factors. Options include o_aitken, o_aitchisonaitken, o_habbema, o_wangvanryzin, and o_liracine. Note that if using np for bw selection above, ordinal kernel types are restricted to either o_wangvanryzin or o_liracine. Defaults to o_wangvanryzin. See details.
stan	a logical value which specifies whether to scale the resulting distance matrix between 0 and 1 using min-max normalization. If set to FALSE, distances are unscaled. Defaults to TRUE.
verbose	a logical value which specifies whether to print procedural steps to the console. If set to FALSE, no output is printed to the console. Defaults to FALSE.

Details

dkss implements the distance using summation similarity distance (DKSS) as described by Ghashti (2024). This approach uses summation kernels for continuous, nominal and ordinal data, which are then summed over all variable types to return the pairwise distance between mixed-type data.

There are several kernels to select from. The continuous kernel functions may be found in Cameron and Trivedi (2005), Härdle et al. (2004) or Silverman (1986). Nominal kernels use a variation on Aitchison and Aitken's (1976) kernel, while ordinal kernels use a variation of the Wang and van Ryzin (1981) kernel. Both nominal and ordinal kernel functions can be found in Li and Racine (2007), Li and Racine (2003), Ouyan et al. (2006), and Titterington and Bowman (1985).

Each kernel requires a bandwidth specification, which can either be a user defined numeric vector of length p from alternative methodologies for bandwidth selection, or through two bandwidth selection methods can be specified. The mscv bandwidth selection is based on maximum similarity cross-validation by Ghashti and Thompson (2024), invoked by the function mscv.dkss. The np bandwidth selection follows the maximum likelihood cross-validation method described by Li and Racine (2007) and Li and Racine (2003) for kernel density estimation of mixed-type data.

Data contained in the data frame df may constitute any combinations of continuous, nominal, or ordinal data, which is to be specified in the data frame df using factor for nominal data, and ordered for ordinal data. Data types can be in any order and will be detected automatically. User-inputted vectors of bandwidths bw must be specified in the same order as the variables in the data frame df, as to ensure they sorted accordingly by the routine.

Value

dkss returns a list object, with the following components:

distances	an $n \times n$ numeric matrix containing pairwise distances between observations
bandwidths	a p-variate vector of bandwidth values returned based on the bw bandwidth specification method, sorted by variable type

Author(s)

John R. J. Thompson <john.thompson@ubc.ca>, Jesse S. Ghashti <jesse.ghashti@ubc.ca>

References

Aitchison, J. and C.G.G. Aitken (1976), "Multivariate binary discrimination by the kernel method," Biometrika, 63, 413-420.

Cameron, A. and P. Trivedi (2005), "Microeconometrics: Methods and Applications", Cambridge University Press.

Ghashti, J.S. (2024), *Similarity Maximization and Shrinkage Approach in Kernel Metric Learning for Clustering Mixed-type Data (T)*, University of British Columbia.

Härdle, W., and M. Müller and S. Sperlich and A. Werwatz (2004), *Nonparametric and Semiparametric Models*, (Vol. 1). Berlin: Springer.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Li, Q. and J.S. Racine (2003), "Nonparametric estimation of distributions with categorical and continuous data," Journal of Multivariate Analysis, 86, 266-292.

Ouyang, D. and Q. Li and J.S. Racine (2006), "Cross-validation and the estimation of probability distributions with categorical data," Journal of Nonparametric Statistics, 18, 69-100.

Silverman, B.W. (1986), *Density Estimation,* London: Chapman and Hall.

Titterington, D.M. and A.W. Bowman (1985), "A comparative study of smoothing procedures for ordered categorical data", Journal of Statistical

Computation and Simulation, 21(3-4), 291-312.

Wang, M.C. and J. van Ryzin (1981), "A class of smooth estimators for discrete distributions," Biometrika, 68, 301-309.

See Also

`mscv.dkps`, `dkps`, `mscv.dkss`

Examples

```
# example data frame with mixed numeric, nominal, and ordinal data.
levels = c("Low", "Medium", "High")
df <- data.frame(
  x1 = runif(100, 0, 100),
  x2 = factor(sample(c("A", "B", "C"), 100, TRUE)),
  x3 = factor(sample(c("A", "B", "C"), 100, TRUE)),
  x4 = rnorm(100, 10, 3),
  x5 = ordered(sample(c("Low", "Medium", "High"), 100, TRUE), levels = levels),
  x6 = ordered(sample(c("Low", "Medium", "High"), 100, TRUE), levels = levels))

# minimal implementation requires just the data frame, and will automatically be
# defaulted to the mscv bandwidth specification technique and default kernel
# function
d1 <- dkss(df = df)
# d$bandwidths to see the mscv obtained bandwidths
# d$distances to see the distance matrix


# try using the np package, which has few continuous and ordinal kernels
```

```
# to choose from. Recommended using default kernel functions
d2 <- dkss(df = df, bw = "np")


# precomputed bandwidth example
# note that continuous variables requires bandwidths > 0
# ordinal variables requires bandwidths in [0,1]
# for nominal variables, u_aitken requires bandwidths in [0,1]
# and u_aitchisonaitken in [0,(c-1)/c]
# where c is the number of unique values in the i-th column of df.
# any bandwidths outside this range will result in a warning message
bw_vec <- c(1.0, 0.5, 0.5, 5.0, 0.3, 0.3)
d3 <- dkss(df = df, bw = bw_vec)


# user-specific kernel functions example
d5 <- dkss(df = df, bw = "mscv", cFUN = "c_epanechnikov", uFUN = "u_aitken",
      oFUN = "o_habbema")
```

kdml *Kernel Metric Learning for Mixed-type Data*

Description

This package contains nonparametric kernel methods for calculating pairwise distances between mixed-type observations. These methods can be used in any distance based algorithm, with emphasis placed on usage in clustering or classification applications. Descriptions of the implementation of these methods can be found in Ghashti (2024) and Ghashti and Thompson (2024).

Details

This package contains two functions for pairwise distance calculations of mixed-type data based on two different methods. Kernel methods also require variable-specific bandwidths, with two additional functions for the bandwidth specification methods. Additionally, this package contains a function methods for mixed-type data generation.

Author(s)

John R.J. Thompson <john.thompson@ubc.ca>, Jesse S. Ghashti <jesse.ghashti@ubc.ca>

Maintainer: John R.J. Thompson <john.thompson@ubc.ca>

We would like to acknowledge funding support from the University of British Columbia Aspire Fund (UBC:www.ok.ubc.ca/). We also acknowledge support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Ghashti, J.S. (2024), *Similarity Maximization and Shrinkage Approach in Kernel Metric Learning for Clustering Mixed-type Data (T)*, University of British Columbia. <https://dx.doi.org/10.14288/1.044397>

Ghashti, J.S. and J.R.J Thompson (2024), "Mixed-type Distance Shrinkage and Selection for Clustering via Kernel Metric Learning." Journal of Classification, Accepted.

`mscv.dkps` *Maximum-similarity Cross-validated (MSCV) bandwidth selection method for the Distance using Kernel Product Similarities (DKPS)*

Description

This function calculates maximum-similarity cross-validated bandwidths for the distance using kernel summation similarity. This implementation uses the method described in Ghashti and Thompson (2024) for mixed-type data that includes any of numeric (continuous), factor (nominal), and ordered factor (ordinal) variables. `mscv.dkps` calculates the bandwidths associated with each kernel function for variable types and returns a numeric vector of bandwidths that can be used with the `dkps` pairwise distance calculation.

Usage

```
mscv.dkps(df, nstart = NULL, ckernel = "c_gaussian", ukernel = "u_aitken",
          okernel = "o_wangvanryzin", verbose = FALSE)
```

Arguments

`df`	a p-variate data frame. The data types may be continuous (`numeric`), nominal (`factor`), ordinal (`ordered`), or any combination thereof. Columns of `df` should be set to the appropriate data type class.
`nstart`	integer number of restarts for the process of finding extrema of the mscv function from random initial bandwidth parameters (starting points). If the default of `NULL` is used, then the number of restarts will be $min(3, \text{ncol(df)})$.
`ckernel`	character string specifying the continuous kernel function. Options include `c_gaussian`, `c_epanechnikov`, `c_uniform`, `c_triangle`, `c_biweight`, `c_triweight`, `c_tricube`, `c_cosine`, `c_logistic`, `c_sigmoid`, and `c_silverman`. Note that if using `np` for `bw` selection above, continuous kernel types are restricted to either `c_gaussian`, `c_epanechnikov`, or `c_uniform`. Defaults to `c_gaussian`. See details.
`ukernel`	character string specifying the nominal kernel function for unordered factors. Options include `u_aitken` and `u_aitchisonaitken`. Defaults to `u_aitken`. See details.
`okernel`	character string specifying the ordinal kernel function for ordered factors. Options include `o_aitken`, `o_aitchisonaitken`, `o_habbema`, `o_wangvanryzin`, and `o_liracine`. Note that if using `np` for `bw` selection above, ordinal kernel types are restricted to either `o_wangvanryzin` or `o_liracine`. Defaults to `o_wangvanryzin`. See details.

`verbose` a logical value which specifies whether to output the i-th iteration of the total number of `nstarts`, and output if the optimization procedure converges. Defaults to `FALSE`.

Details

`mscv.dkps` implements the maximum-similarity cross-validation (MSCV) technique for bandwidth selection pertaining to the `dkps` function, as described by Ghashti and Thompson (2024). This approach uses product kernels for continuous variables, and summation kernels for nominal and ordinal data, which are then summed over all variable types to return the pairwise distance between mixed-type data.

The maximization procedure for bandwidth selection is based on the objective $\arg\max_{\boldsymbol{\lambda}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \log \left(\frac{1}{(n-1)} \sum_{\substack{j=1 \\ j \neq i}}^{n} \psi_{\boldsymbol{\lambda}}(\mathbf{x}_i, \right. \right.$ where

$\psi(\mathbf{x}_i, \mathbf{x}_j \,|\boldsymbol{\lambda}) = \prod_{k=1}^{p_c} \frac{1}{\lambda_k^c} K(x_{i,k}^c, x_{j,k}^c, \lambda_k^c) + \sum_{k=1}^{p_u} L(x_{i,k}^u, x_{j,k}^u, \lambda_k^u) + \sum_{k=1}^{p_o} \ell(x_{i,k}^o, x_{j,k}^o, \lambda_k^o).$

$K(\cdot)$, $L(\cdot)$, and $\ell(\cdot)$ are the continuous, nominal, and ordinal kernel functions, repectively, with λ_k's representing kernel specifical bandwiths for the k-th variable, and p_c, p_u, p_o the number of continuous, nominal, and ordinal variables in the data frame `df`. The resulting `bw` vector returned is the bandwidths that yield the highest objective function value.

Data contained in the data frame `df` may constitute any combinations of continuous, nominal, or ordinal data, which is to be specified in the data frame `df` using `numeric` for continuous data, `factor` for nominal data, and `ordered` for ordinal data. Data can be entered in an arbitrary order and data types will be detected automatically. User-inputted vectors of bandwidths `bw` must be defined in the same order as the variables in the data frame `df`, as to ensure they sorted accordingly by the routine.

The are many kernels which can be specified by the user. Continuous kernel functions may be found in Cameron and Trivedi (2005), Härdle et al. (2004) or Silverman (1986). Nominal kernels use a variation on Aitchison and Aitken's (1976) kernel. Ordinal kernels use a variation of the Wang and van Ryzin (1981) kernel. All nominal and ordinal kernel functions can be found in Li and Racine (2007), Li and Racine (2003), Ouyan et al. (2006), and Titterington and Bowman (1985).

Value

`mscv.dkps` returns a `list` object, with the following components:

`bw` a p-variate vector of bandwidth values, intended to be used for the `dkps` pairwise distance calculation

`fn_value` a numeric value of the MSCV objective function, obtained using the `optim` function for constrained multivariate optimization

Author(s)

John R. J. Thompson <john.thompson@ubc.ca>, Jesse S. Ghashti <jesse.ghashti@ubc.ca>

References

Aitchison, J. and C.G.G. Aitken (1976), "Multivariate binary discrimination by the kernel method," Biometrika, 63, 413-420.

Cameron, A. and P. Trivedi (2005), “Microeconometrics: Methods and Applications”, Cambridge University Press.

Ghashti, J.S. and J.R.J Thompson (2024), “Mixed-type Distance Shrinkage and Selection for Clustering via Kernel Metric Learning.” Journal of Classification, Accepted.

Härdle, W., and M. Müller and S. Sperlich and A. Werwatz (2004), *Nonparametric and Semiparametric Models*, (Vol. 1). Berlin: Springer.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Li, Q. and J.S. Racine (2003), “Nonparametric estimation of distributions with categorical and continuous data,” Journal of Multivariate Analysis, 86, 266-292.

Ouyang, D. and Q. Li and J.S. Racine (2006), “Cross-validation and the estimation of probability distributions with categorical data,” Journal of Nonparametric Statistics, 18, 69-100.

Silverman, B.W. (1986), *Density Estimation,* London: Chapman and Hall.

Titterington, D.M. and A.W. Bowman (1985), “A comparative study of smoothing procedures for ordered categorical data”, Journal of Statistical Computation and Simulation, 21(3-4), 291-312.

Wang, M.C. and J. van Ryzin (1981), “A class of smooth estimators for discrete distributions,” Biometrika, 68, 301-309.

See Also

mscv.dkss, dkss, dkps

Examples

```
# example data frame with mixed numeric, nominal, and ordinal data.
levels = c("Low", "Medium", "High")
df <- data.frame(
  x1 = runif(100, 0, 100),
  x2 = factor(sample(c("A", "B", "C"), 100, TRUE)),
  x3 = factor(sample(c("A", "B", "C"), 100, TRUE)),
  x4 = rnorm(100, 10, 3),
  x5 = ordered(sample(c("Low", "Medium", "High"), 100, TRUE), levels = levels),
  x6 = ordered(sample(c("Low", "Medium", "High"), 100, TRUE), levels = levels))

# minimal implementation requires just the data frame, with defaults
bw <- mscv.dkps(df = df)

# specify number of starts and kernel functions
bw2 <- mscv.dkps(df = df, nstart = 5, ckernel = "c_triangle",
                  ukernel = "u_aitken", okernel = "o_liracine")
```

mscv.dkss *Maximum-similarity Cross-validated (MSCV) bandwidth selection method for the distance using kernel summation similarity (DKSS)*

Description

This function calculates maximum-similarity cross-validated bandwidths for the distance using kernel summation similarity. This implementation uses the method described in Ghashti (2024) for mixed-type data that includes any of numeric (continuous), factor (nominal), and ordered factor (ordinal) variables. `mscv.dkss` calculates the bandwidths associated with each kernel function for variable types and returns a numeric vector of bandwidths that can be used with the `dkss` pairwise distance calculation.

Usage

```
mscv.dkss(df, nstart = NULL, ckernel = "c_gaussian", ukernel = "u_aitken",
          okernel = "o_wangvanryzin", verbose = FALSE)
```

Arguments

`df` a p-variate data frame. The data types may be continuous (`numeric`), nominal (`factor`), ordinal (`ordered`), or any combination thereof. Columns of `df` should be set to the appropriate data type class.

`nstart` integer number of restarts for the process of finding extrema of the mscv function from random initial bandwidth parameters (starting points). If the default of `NULL` is used, then the number of restarts will be $min(3, \text{ncol(df)})$.

`ckernel` character string specifying the continuous kernel function. Options include `c_gaussian`, `c_epanechnikov`, `c_uniform`, `c_triangle`, `c_biweight`, `c_triweight`, `c_tricube`, `c_cosine`, `c_logistic`, `c_sigmoid`, and `c_silverman`. Note that if using `np` for `bw` selection above, continuous kernel types are restricted to either `c_gaussian`, `c_epanechnikov`, or `c_uniform`. Defaults to `c_gaussian`. See details.

`ukernel` character string specifying the nominal kernel function for unordered factors. Options include `u_aitken` and `u_aitchisonaitken`. Defaults to `u_aitken`. See details.

`okernel` character string specifying the ordinal kernel function for ordered factors. Options include `o_aitken`, `o_aitchisonaitken`, `o_habbema`, `o_wangvanryzin`, and `o_liracine`. Note that if using `np` for `bw` selection above, ordinal kernel types are restricted to either `o_wangvanryzin` or `o_liracine`. Defaults to `o_wangvanryzin`. See details.

`verbose` a logical value which specifies whether to output the i-th iteration of the total number of `nstarts`, and output if the optimization procedure converges. Defaults to `FALSE`.

Details

`mscv.dkss` implements the maximum-similarity cross-validation (MSCV) bandwidth selection technique for the `dkss` function, described by Ghashti (2024). This approach uses summation kernels for continuous, nominal and ordinal data, which are then summed over all variable types to return the pairwise distance between mixed-type data.

The maximization procedure for bandwidth selection is based on the objective $\arg\max_{\boldsymbol{\lambda}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \log \left(\frac{1}{(n-1)} \sum_{\substack{j=1 \\ j \neq i}}^{n} s_{\text{KSS}_{\boldsymbol{\lambda}}}(\mathbf{x}\right.\right.$ where

$s_{\text{KSS}}(\mathbf{x}_i, \mathbf{x}_j \,|\boldsymbol{\lambda}) = \sum_{k=1}^{p_c} K(x_{i,k}^c, x_{j,k}^c, \lambda_k^c) + \sum_{k=1}^{p_u} L(x_{i,k}^u, x_{j,k}^u, \lambda_k^u) + \sum_{k=1}^{p_o} \ell(x_{i,k}^o, x_{j,k}^o, \lambda_k^o)$. $K(\cdot)$, $L(\cdot)$, and $\ell(\cdot)$ are the continuous, nominal, and ordinal kernel functions, repectively, with λ_k's representing kernel specifical bandwiths for the k-th variable, and p_c, p_u, p_o the number of continuous, nominal, and ordinal variables in the data frame `df`. The `bw` vector returned is the bandwidths that yield the highest objective function value.

Data contained in the data frame `df` may constitute any combinations of continuous, nominal, or ordinal data, which is to be specified in the data frame `df` using `numeric` for continuous data, `factor` for nominal data, and `ordered` for ordinal data. Data can be entered in an arbitrary order and data types will be detected automatically. User-inputted vectors of bandwidths `bw` must be defined in the same order as the variables in the data frame `df`, as to ensure they sorted accordingly by the routine.

The are many kernels which can be specified by the user. Continuous kernel functions may be found in Cameron and Trivedi (2005), Härdle et al. (2004) or Silverman (1986). Nominal kernels use a variation of Aitchison and Aitken's (1976) kernel. Ordinal kernels use a variation of the Wang and van Ryzin (1981) kernel. All nominal and ordinal kernel functions can be found in Li and Racine (2007), Li and Racine (2003), Ouyan et al. (2006), and Titterington and Bowman (1985).

Value

`mscv.dkss` returns a `list` object, with the following components:

`bw`	a p-variate vector of bandwidth values, intended to be used for the `dkss` pairwise distance calculation
`fn_value`	a numeric value of the MSCV objective function, obtained using the `optim` function for constrained multivariate optimization

Author(s)

John R. J. Thompson <john.thompson@ubc.ca>, Jesse S. Ghashti <jesse.ghashti@ubc.ca>

References

Aitchison, J. and C.G.G. Aitken (1976), "Multivariate binary discrimination by the kernel method," Biometrika, 63, 413-420.

Cameron, A. and P. Trivedi (2005), "Microeconometrics: Methods and Applications", Cambridge University Press.

Ghashti, J.S. (2024), *Similarity Maximization and Shrinkage Approach in Kernel Metric Learning for Clustering Mixed-type Data*, University of British Columbia.

Härdle, W., and M. Müller and S. Sperlich and A. Werwatz (2004), *Nonparametric and Semiparametric Models*, (Vol. 1). Berlin: Springer.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Li, Q. and J.S. Racine (2003), "Nonparametric estimation of distributions with categorical and continuous data," Journal of Multivariate Analysis, 86, 266-292.

Ouyang, D. and Q. Li and J.S. Racine (2006), "Cross-validation and the estimation of probability distributions with categorical data," Journal of Nonparametric Statistics, 18, 69-100.

Silverman, B.W. (1986), *Density Estimation,* London: Chapman and Hall.

Titterington, D.M. and A.W. Bowman (1985), "A comparative study of smoothing procedures for ordered categorical data", Journal of Statistical Computation and Simulation, 21(3-4), 291-312.

Wang, M.C. and J. van Ryzin (1981), "A class of smooth estimators for discrete distributions," Biometrika, 68, 301-309.

See Also

`mscv.dkps`, `dkss`, `dkps`

Examples

```
# example data frame with mixed numeric, nominal, and ordinal data.
levels = c("Low", "Medium", "High")
df <- data.frame(
  x1 = runif(100, 0, 100),
  x2 = factor(sample(c("A", "B", "C"), 100, TRUE)),
  x3 = factor(sample(c("A", "B", "C"), 100, TRUE)),
  x4 = rnorm(100, 10, 3),
  x5 = ordered(sample(c("Low", "Medium", "High"), 100, TRUE), levels = levels),
  x6 = ordered(sample(c("Low", "Medium", "High"), 100, TRUE), levels = levels))

# minimal implementation requires just the data frame, with defaults
bw <- mscv.dkss(df = df)

# specify number of starts and kernel functions
bw2 <- mscv.dkss(df = df, nstart = 5, ckernel = "c_triangle",
                ukernel = "u_aitken", okernel = "o_liracine")
```

Index

∗ **~distances**
  dkps, 4
  dkss, 7
  mscv.dkps, 11
  mscv.dkss, 14
∗ **~metriclearning**
  dkps, 4
  dkss, 7
  mscv.dkps, 11
  mscv.dkss, 14
∗ **~metrics**
  dkps, 4
  dkss, 7
  mscv.dkps, 11
  mscv.dkss, 14
∗ **~multivariate**
  dkps, 4
  dkss, 7
  mscv.dkps, 11
  mscv.dkss, 14
∗ **~nonparametric**
  dkps, 4
  dkss, 7
  mscv.dkps, 11
  mscv.dkss, 14
∗ **~optimize**
  dkps, 4
  dkss, 7
  mscv.dkps, 11
  mscv.dkss, 14
∗ **package**
  kdml, 10

confactord, 2

data.frame, *3*
dkps, *3*, 4, *9*, *12*, *13*, *16*
dkss, *3*, *6*, 7, *13*, *15*, *16*

factor, *5*, *8*, *11*, *12*, *14*, *15*

kdml, 10
kdml-package *(*kdml*)*, 10

mscv.dkps, *3*, *6*, *9*, 11, *16*
mscv.dkss, *3*, *5*, *6*, *8*, *9*, *13*, 14

np, *4*, *7*
npudensbw, *4*, *7*
numeric, *11*, *12*, *14*, *15*

optim, *12*, *15*
ordered, *5*, *8*, *11*, *12*, *14*, *15*