Package ‘kerntools’

February 19, 2025

Type Package

Title Kernel Functions and Tools for Machine Learning Applications

Version 1.2.0

Description Kernel functions for diverse types of data (including, but not restricted to: nonnegative and real vectors, real matrices, categorical and ordinal variables, sets, strings), plus other utilities like kernel similarity, kernel Principal Components Analysis (PCA) and features' importance for Support Vector Machines (SVMs), which expand other 'R' packages like 'kernlab'.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, ggplot2, kernlab, methods, reshape2, stats, stringi

VignetteBuilder knitr

Depends R (>= 2.10)

Language en-US

URL https://github.com/elies-ramon/kerntools, https://elies-ramon.github.io/kerntools/

BugReports https://github.com/elies-ramon/kerntools/issues

NeedsCompilation no

Author Elies Ramon [aut, cre, cph] (<https://orcid.org/0000-0002-7953-8115>)

Maintainer Elies Ramon <eramon@everlyrusher.com>

Repository CRAN

Date/Publication 2025-02-19 15:10:02 UTC

Contents

Acc *Accuracy*

Description

‘Acc()‘ computes the accuracy between the output of a classification model and the actual values of the target. It can also compute the weighted accuracy, which is useful in imbalanced classification problems. The weighting is applied according to the class frequencies in the target. In balanced problems, weighted Acc = Acc.

Usage

```
Acc(ct, weighted = FALSE)
```

Arguments

`ct` Confusion Matrix.

`weighted` If TRUE, the weighted accuracy is returned. (Defaults: FALSE).

Value

Accuracy of the model (a single value).

Examples

```
y <- c(rep("a",3),rep("b",2))
y_pred <- c(rep("a",2),rep("b",3))
ct <- table(y,y_pred)
Acc(ct)
Acc(ct,weighted=TRUE)
```

Acc_rnd *Accuracy of a random model*

Description

‘Acc_rnd()‘ computes the expected accuracy of a random classifier based on the class frequencies of the target. This measure can be used as a benchmark when contrasted to the accuracy (in test) of a given prediction model.

Usage

```
Acc_rnd(target, freq = FALSE)
```

Arguments

`target` A character vector or a factor. Alternatively, a numeric vector (see below).

`freq` TRUE if ‘target‘ contains the frequencies of the classes (in this case, ‘target‘ should be numeric), FALSE otherwise. (Defaults: FALSE).

Value

Expected accuracy of a random classification model (a single value).

Examples

```
# Expected accuracy of a random model:
target <- c(rep("a",5),rep("b",2))
Acc_rnd(target)
# This is the same than:
freqs <- c(5/7,2/7)
Acc_rnd(freqs,freq=TRUE)
```

`aggregate_imp` *Aggregate importances*

Description

‘aggregate_imp()‘ sums the importances present in a matrix or data.frame according to some user-specified grouping criterion.

Usage

```
aggregate_imp(X, lev = NULL, samples = "rows")
```

Arguments

`X` Matrix or data.frame containing the importances (in rows or in columns).

`lev` (optional) The grouping elements. ‘lev‘ should be as long as the dimension (cols or rows) that one wants to aggregate. If this parameter is absent, the colnames (if samples="rows") or rownames will be used to that effect. In that case, it is expected that the col/rownames follow this pattern: "V_Y", and the variables with the same "V" will be summed. (Check the colnames of a typical output of ‘dummy_data()‘ for more info).

`samples` Samples are in rows or in columns? (Defaults: "rows").

Value

X, a matrix or data.frame containing the aggregated importances.

Examples

```
importances <- matrix(rnorm(90),nrow=3,ncol=30)
rownames(importances) <- c("sample1","sample2","sample3")
colnames(importances) <- paste0("Feat",
rep(1:5,times=2*(1:5)), "_", unlist(lapply(2*(1:5),function(x)LETTERS[1:x])))

## The grouping criterion is:
groups <- paste0("Feat",1:5)
aggregate_imp(X=importances,samples="rows",lev=groups)
## We can also use the colnames:
colnames(importances)
aggregate_imp(X=importances,samples="rows")
```

`Boots_CI` *Confidence Interval using Bootstrap*

Description

‘Boots_CI()‘ computes the Confidence Interval (CI) of a performance measure (for instance, accuracy) via bootstrapping.

Usage

```
Boots_CI(target, pred, index = "acc", nboots, confidence = 95, ...)
```

Arguments

`target` Numeric vector containing the actual values.

`pred` Numeric vector containing the predicted values. (The order should be the same than the target’s).

`index` Performance measure name, in lowercase. (Defaults: "acc").

`nboots` Number of bootstrapping replicas.

`confidence` Confidence level; for instance, 95% or 99%. (Defaults: 95).

`...` Further arguments to be passed to the performance measures functions; notably, multi.class="macro" or multi.class="micro" for the macro or micro performance measures. (Defaults: "macro").

Value

A vector containing the bootstrap estimate of the performance and its CI.

Examples

```
y <- c(rep("a",30),rep("b",20))
y_pred <- c(rep("a",20),rep("b",30))
# Computing Accuracy with their 95%CI
Boots_CI(target=y, pred=y_pred, index="acc", nboots=1000, confidence=95)
```

BrayCurtis *Kernels for count data*

Description

Ruzicka and Bray-Curtis are kernel functions for absolute or relative frequencies and count data. Both kernels have as input a matrix or data.frame with dimension *NxD* and *N*>1, *D*>1, containing strictly non-negative real numbers. Samples should be in the rows. Thus, when working with relative frequencies, ‘rowSums(X)‘ should be 1 (or 100, or another arbitrary number) for *all* rows (samples) of the dataset.

Usage

```
BrayCurtis(X)

Ruzicka(X)
```

Arguments

`X`	Matrix or data.frame that contains absolute or relative frequencies.

Details

For more info about these measures, please check Details in ?vegan::vegdist(). Note that, in the vegan help page, "Ruzicka" corresponds to "quantitative Jaccard". ‘BrayCurtis(X)‘ gives the same result than ‘1-vegan::vegdist(X,method="bray")‘, and the same with ‘Ruzicka(data)‘ and ‘1-vegan::vegdist(data,method="jaccard")‘.

Value

Kernel matrix (dimension: *NxN*).

Examples

```
data <- soil$abund
Kruz <- Ruzicka(data)
Kbray <- BrayCurtis(data)
Kruz[1:5,1:5]
Kbray[1:5,1:5]
```

centerK *Centering a kernel matrix*

Description

It is equivalent to compute ‘K‘ over centered data (i.e. the mean of each column is subtracted) in Feature Space.

Usage

```
centerK(K)
```

Arguments

K Kernel matrix (class "matrix").

Value

Centered ‘K‘ (class "matrix").

Examples

```
dat <- matrix(rnorm(250),ncol=50,nrow=5)
K <- Linear(dat)
centerK(K)
```

centerX *Centering a squared matrix by row or column*

Description

It centers a numeric matrix with dimension *N x N* by row (rows=TRUE) or column (rows=FALSE).

Usage

```
centerX(X, rows = TRUE)
```

Arguments

X Numeric matrix or data.frame of any size.

rows If TRUE, the operation is done by row; otherwise, it is done by column. (Defaults: TRUE).

Value

Centered X (class "matrix").

Examples

```
dat <- matrix(rnorm(25),ncol=5,nrow=5)
centerX(dat)
```

Chi2 *Chi-squared kernel*

Description

'Chi2()' computes the basic χ^2 kernel for bag-of-words (BoW) or bag-of-visual-words data. This kernel computes the similarity between two nonnegative vectors that represent the occurrence counts of words in two different documents.

Usage

```
Chi2(X, g = NULL)
```

Arguments

X	Matrix or data.frame (dimension *NxD*) that contains nonnegative numbers. Each row represents the counts of words of *N* documents, while each column is a word.
g	Gamma hyperparameter. If g=0 or NULL, 'Chi2()' returns the LeCam distances between the documents instead of the χ^2 kernel matrix. (Defaults=NULL).

Value

Kernel matrix (dimension: *NxN*).

References

Zhang, Jianguo, et al. Local features and kernels for classification of texture and object categories: A comprehensive study. International journal of computer vision 73 (2007): 213-238. Link

Examples

```
## Example dataset: word counts in 4 documents
documents <- matrix( c(0, 1, 3, 2, 1, 0,  1, 1, 6,4,3,1,3,5,6,2), nrow=4,byrow=TRUE)
rownames(documents) <- paste0("doc",1:4)
colnames(documents) <- c("animal","life","tree","ecosystem")
documents
Chi2(documents,g=NULL)
```

cLinear *Compositional kernels*

Description

‘cLinear()‘ is the compositional-linear kernel, which is useful for compositional data (relative frequencies or proportions). ‘Aitchison()‘ is akin to the RBF kernel for this type of data. Thus, the expected input for both kernels is a matrix or data.frame containing strictly non-negative or (even better) positive numbers. This input has dimension *NxD*, with *N*>1 samples and *D*>1 compositional features.

Usage

```
cLinear(X, cos.norm = FALSE, feat_space = FALSE, zeros = "none")

Aitchison(X, g = NULL, zeros = "none")
```

Arguments

`X`	Matrix or data.frame that contains the compositional data.
`cos.norm`	Should the resulting kernel matrix be cosine normalized? (Defaults: FALSE).
`feat_space`	If FALSE, only the kernel matrix is returned. Otherwise, the feature space is also returned. (Defaults: FALSE).
`zeros`	"none" to warrant that there are no zeroes in X, "pseudo" to replace zeroes by a pseudocount. (Defaults="none").
`g`	Gamma hyperparameter. If g=0 or NULL, the matrix of squared Aitchison distances is returned instead of the Aitchison kernel matrix. (Defaults=NULL).

Details

In compositional data, samples (rows) sum to an arbitrary or irrelevant number. This is most clear when working with relative frequencies, as all samples add to 1 (or 100, or other uninformative value). Zeroes are a typical challenge when using compositional approaches. They introduce ambiguity because they can have multiple causes; a zero may signal a true absence, or a value so small that it is below the detection threshold of an instrument. A simple approach to deal with zeroes is replacing them by a pseudocount. More sophisticated approaches are reviewed elsewhere; see for instance the R package ‘zCompositions‘.

Value

Kernel matrix (dimension: *NxN*).

References

Ramon, E., Belanche-Muñoz, L. et al (2021). kernInt: A kernel framework for integrating supervised and unsupervised analyses in spatio-temporal metagenomic datasets. Frontiers in microbiology 12 (2021): 609048. doi: 10.3389/fmicb.2021.609048

Examples

```
data <- soil$abund

## This data is sparse and contains a lot of zeroes. We can replace them by pseudocounts:
Kclin <- cLinear(data,zeros="pseudo")
Kclin[1:5,1:5]

## With the feature space:
Kclin <- cLinear(data,zeros="pseudo",feat_space=TRUE)

## With cosine normalization:
Kcos <- cLinear(data,zeros="pseudo",cos.norm=TRUE)
Kcos[1:5,1:5]

## Aitchison kernel:
Kait <- Aitchison(data,g=0.0001,zeros="pseudo")
Kait[1:5,1:5]
```

---

cosNorm *Cosine normalization of a kernel matrix*

Description

It is equivalent to compute K using the normalization 'X/sqrt(sum(X^2))' in Feature Space.

Usage

```
cosNorm(K)
```

Arguments

K	Kernel matrix (class "matrix").

Value

Cosine-normalized K (class "matrix").

References

Ah-Pine, J. (2010). Normalized kernels as similarity indices. In Advances in Knowledge Discovery and Data Mining: 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II 14 (pp. 362-373). Springer Berlin Heidelberg. Link

Examples

```
dat <- matrix(rnorm(250),ncol=50,nrow=5)
K <- Linear(dat)
cosNorm(K)
```

`cosnormX` *Cosine normalization of a matrix*

Description

Normalizes a numeric matrix dividing each row (if rows=TRUE) or column (if rows=FALSE) by their L2 norm. Thus, each row (or column) has unit norm.

Usage

```
cosnormX(X, rows = TRUE)
```

Arguments

`X` Numeric matrix or data.frame of any size.

`rows` If TRUE, the operation is done by row; otherwise, it is done by column. (Defaults: TRUE).

Value

Cosine-normalized X.

Examples

```
dat <- matrix(rnorm(50),ncol=5,nrow=10)
cosnormX(dat)
```

`desparsify` *This function deletes those columns and/or rows in a matrix/data.frame that only contain 0s.*

Description

This function deletes those columns and/or rows in a matrix/data.frame that only contain 0s.

Usage

```
desparsify(X, dim = 2)
```

Arguments

`X` Numeric matrix or data.frame of any size.

`dim` A numeric vector. 1 indicates that the function should be applied to rows, 2 to columns, c(1, 2) indicates rows and columns. (Defaults: 2).

Value

X with less rows or columns. (Class: the same than X).

Examples

```
dat <- matrix(rnorm(150),ncol=50,nrow=30)
dat[c(2,6,12),] <- 0
dat[,c(30,40,50)] <- 0
dim(desparsify(dat))
dim(desparsify(dat,dim=c(1,2)))
```

Dirac *Kernels for categorical variables*

Description

From a matrix or data.frame with dimension *NxD*, where *N*>1, *D*>0, ‘Dirac()‘ computes the simplest kernel for categorical data. Samples should be in the rows and features in the columns. When there is a single feature, ‘Dirac()‘ returns 1 if the category (or class, or level) is the same in two given samples, and 0 otherwise. Instead, when *D*>1, the results for the *D* features are combined doing a sum, a mean, or a weighted mean.

Usage

```
Dirac(X, comp = "mean", coeff = NULL, feat_space = FALSE)
```

Arguments

`X` Matrix (class "character") or data.frame (class "character", or columns = "factor"). The elements in X are assumed to be categorical in nature.

`comp` When *D*>1, this argument indicates how the variables of the dataset are combined. Options are: "mean", "sum" and "weighted". (Defaults: "mean")

- "sum" gives the same importance to all variables, and returns an unnormalized kernel matrix.
- "mean" gives the same importance to all variables, and returns a normalized kernel matrix (all its elements range between 0 and 1).
- "weighted" weights each variable according to the ‘coeff‘ parameter, and returns a normalized kernel matrix.

`coeff` (optional) A vector of weights with length *D*.

`feat_space` If FALSE, only the kernel matrix is returned. Otherwise, the feature space is also returned. (Defaults: FALSE).

Value

Kernel matrix (dimension: *NxN*), or a list with the kernel matrix and the feature space.

References

Belanche, L. A., and Villegas, M. A. (2013). Kernel functions for categorical variables with application to problems in the life sciences. Artificial Intelligence Research and Development (pp. 171-180). IOS Press. Link

Examples

```
# Categorical data
summary(CO2)
Kdirac <- Dirac(CO2[,1:3])
## Display a subset of the kernel matrix:
Kdirac[c(1,15,50,65),c(1,15,50,65)]
```

---

dummy_data *Convert categorical data to dummies.*

Description

Given a matrix or data.frame containing character/factors, this function performs one-hot-encoding.

Usage

```
dummy_data(X, lev = NULL)
```

Arguments

X	A matrix, or a data.frame containing factors. (If the columns are of any other class, they will be coerced into factors anyway).
lev	(optional) A vector with the categories ("levels") of each factor.

Value

X (class: "matrix") after performing one-hot-encoding.

Examples

```
summary(CO2)
CO2_dummy <- dummy_data(CO2[,1:3],lev=dummy_var(CO2[,1:3]))
CO2_dummy[1:10,1:5]
```

`dummy_var` *Levels per factor variable*

Description

This function gives the categories ("levels") per categorical variable ("factor").

Usage

```
dummy_var(X)
```

Arguments

`X`	A matrix, or a data.frame containing factors. (If the columns are of any other class, they will be coerced into factors anyway).

Value

A list with the levels.

Examples

```
summary(showdata)
dummy_var(showdata)
```

`estimate_gamma` *Gamma hyperparameter estimation (RBF kernel)*

Description

This function returns an estimation of the optimum value for the gamma hyperparameter (required by the RBF kernel function) using different heuristics:

***D* criterion** It returns the inverse of the number of features in X.

Scale criterion It returns the inverse of the number of features, normalized by the total variance of X.

Quantiles criterion A range of values, computed with the function ‘kernlab::sigest()‘.

Usage

```
estimate_gamma(X)
```

Arguments

`X`	Matrix or data.frame that contains real numbers ("integer", "float" or "double").

Value

A list with the gamma value estimation according to different criteria.

Examples

```
data <- matrix(rnorm(150),ncol=50,nrow=30)
gamma <- estimate_gamma(data)
gamma
K <- RBF(data, g = gamma$scale_criterion)
K[1:5,1:5]
```

F1 *F1 score*

Description

‘F1()‘ computes the F1 score between the output of a classification prediction model and the actual values of the target.

Usage

```
F1(ct, multi.class = ”macro”)
```

Arguments

ct	Confusion Matrix.
multi.class	Should the results of each class be aggregated, and how? Options: "none", "macro", "micro". (Defaults: "macro").

Details

F1 corresponds to the harmonic mean of Precision and Recall.

Value

F1 (a single value).

Examples

```
y <- c(rep(”a”,3),rep(”b”,2))
y_pred <- c(rep(”a”,2),rep(”b”,3))
ct <- table(y,y_pred)
F1(ct)
```

Frobenius *Frobenius kernel*

Description

‘Frobenius()‘ computes the Frobenius kernel between numeric matrices.

Usage

```
Frobenius(DATA, cos.norm = FALSE, feat_space = FALSE)
```

Arguments

DATA	A list of *M* matrices or data.frames containing only real numbers (class "integer", "float" or "double"). All matrices or data.frames should have the same number of rows and columns.
cos.norm	Should the resulting kernel matrix be cosine normalized? (Defaults: FALSE).
feat_space	If FALSE, only the kernel matrix is returned. Otherwise, the feature space is also returned. (Defaults: FALSE).

Details

The Frobenius kernel is the same than the Frobenius inner product between matrices.

Value

Kernel matrix (dimension:*NxN*), or a list with the kernel matrix and the feature space.

Examples

```
data1 <- matrix(rnorm(250000),ncol=500,nrow=500)
data2 <- matrix(rnorm(250000),ncol=500,nrow=500)
data3 <- matrix(rnorm(250000),ncol=500,nrow=500)

Frobenius(list(data1,data2,data3))
```

frobNorm *Frobenius normalization*

Description

This function computes the Frobenius normalization of a matrix.

Usage

```
frobNorm(X)
```

Arguments

X	Numeric matrix of any size. It may be a kernel matrix.

Value

Frobenius-normalized X (class: "matrix").

Examples

```
dat <- matrix(rnorm(50),ncol=5,nrow=10)
frobNorm(dat)
```

heatK *Kernel matrix heatmap*

Description

‘heatK()‘ plots the heatmap of a kernel matrix.

Usage

```
heatK(
  K,
  cos.norm = FALSE,
  title = NULL,
  color = c("red", "yellow"),
  name_leg = NULL,
  raster = FALSE
)
```

Arguments

K	Kernel matrix (class "matrix").
cos.norm	If TRUE, the cosine normalization is applied to the kernel matrix so its elements have a maximum value of 1. (Defaults: FALSE).
title	Heatmap title (optional).
color	A vector of length 2 containing two colors. The first color will be used to represent the minimum value and the second the maximum value of the kernel matrix.
name_leg	Title of the legend.
raster	In large kernel matrices, raster = TRUE will draw quicker and better-looking heatmaps. (Defaults=FALSE).

Value

A ‘ggplot2‘ heatmap.

Examples

```
data <- matrix(rnorm(150),ncol=50,nrow=30)
K <- Linear(data)
heatK(K)
```

histK *Kernel matrix histogram*

Description

‘histK()‘ plots the histogram of a kernel matrix.

Usage

```
histK(K, main = "Histogram of K", vn = FALSE, ...)
```

Arguments

`K`	Kernel matrix (class "matrix").
`main`	Plot title.
`vn`	If TRUE, the value of the von Neumann entropy is shown in the plot. (Defaults: FALSE).
`...`	further arguments and graphical parameters passed to ‘plot.histogram‘.

Details

Information about the von Neumann entropy can be found at ’?vonNeumann()’.

Value

An object of class "histogram".

Examples

```
data <- matrix(rnorm(150),ncol=50,nrow=30)
K <- RBF(data,g=0.01)
histK(K)
```

Jaccard *Kernels for sets*

Description

‘Intersect()‘ or ‘Jaccard()‘ compute the kernel functions of the same name, which are useful for set data. Their input is a matrix or data.frame with dimension *NxD*, where *N*>1, *D*>0. Samples should be in the rows and features in the columns. When there is a single feature, ‘Jaccard()‘ returns 1 if the elements of the set are exactly the same in two given samples, and 0 if they are completely different (see Details). Instead, in the multivariate case (*D*>1), the results (for both ‘Intersect()‘ and ‘Jaccard()‘) of the *D* features are combined with a sum, a mean, or a weighted mean.

Usage

```
Jaccard(X, elements = LETTERS, comp = "sum", coeff = NULL)

Intersect(
  X,
  elements = LETTERS,
  comp = "sum",
  coeff = NULL,
  feat_space = FALSE
)
```

Arguments

`X` Matrix (class "character") or data.frame (class "character", or columns = "factor"). The elements in X are assumed to be categorical in nature.

`elements` All potential elements (symbols) that can appear in the sets. If there are some elements that are not of interest, they can be excluded so they are not taken into account by these kernels. (Defaults: LETTERS).

`comp` When *D*>1, this argument indicates how the variables of the dataset are combined. Options are: "mean", "sum" and "weighted". (Defaults: "mean")

- "sum" gives the same importance to all variables, and returns an unnormalized kernel matrix.
- "mean" gives the same importance to all variables, and returns a normalized kernel matrix (all its elements range between 0 and 1).
- "weighted" weights each variable according to the ‘coeff‘ parameter, and returns a normalized kernel matrix.

`coeff` (optional) A vector of weights with length *D*.

`feat_space` (not available for the Jaccard kernel). If FALSE, only the kernel matrix is returned. Otherwise, the feature space is returned too. (Defaults: FALSE).

Details

Let A, B be two sets. Then, the Intersect kernel is defined as:

$$K_{Intersect}(A,B) = |A \cap B|$$

And the Jaccard kernel is defined as:

$$K_{Jaccard}(A,B) = |A \cap B|/|A \cup B|$$

This specific implementation of the Intersect and Jaccard kernels expects that the set members (elements) are character symbols (length=1). In case the set data is multivariate (*D*>1 columns, and each one contains a set feature), elements for the *D* sets should come from the same domain (universe). For instance, a dataset with two variables, so the elements in the first one are colors c("green","black","white","red") and the second are names c("Anna","Elsa","Maria") is not allowed. In that case, set factors should be recoded to colors c("g","b","w","r") and names c("A","E","M") and, if necessary, 'Intersect()' (or ‘Jaccard()‘) should be called twice.

Value

Kernel matrix (dimension: *NxN*), or a list with the kernel matrix and the feature space.

References

Bouchard, M., Jousselme, A. L., and Doré, P. E. (2013). A proof for the positive definiteness of the Jaccard index matrix. International Journal of Approximate Reasoning, 54(5), 615-626.

Ruiz, F., Angulo, C., and Agell, N. (2008). Intersection and Signed-Intersection Kernels for Intervals. Frontiers in Artificial Intelligence and Applications. 184. 262-270. doi: 10.3233/978-1-58603-925-7-262.

Examples

```
# Sets data
## Generating a dataset with sets containing uppercase letters
random_set <- function(x)paste(sort(sample(LETTERS,x,FALSE)),sep="",collapse = "")
max_setsize <- 4
setsdata <- matrix(replicate(20,random_set(sample(2:max_setsize,1))),nrow=4,ncol=5)

## Computing the Intersect kernel:
Intersect(setsdata,elements=LETTERS,comp="sum")

## Computing the Jaccard kernel weighting the variables:
coeffs <- c(0.1,0.15,0.15,0.4,0.20)
Jaccard(setsdata,elements=LETTERS,comp="weighted",coeff=coeffs)
```

Kendall *Kendall's tau kernel*

Description

'Kendall()' computes the Kendall's tau, which happens to be a kernel function for ordinal variables, ranks or permutations.

Usage

```
Kendall(X, NA.as.0 = TRUE, samples.in.rows = FALSE, comp = "mean")
```

Arguments

X	When evaluating a single ordinal feature, X should be a numeric matrix or data.frame. If data is multivariate, X should be a list, and each ordinal/ranking feature should be placed in a different element of the list (see Examples).
NA.as.0	Should NAs be converted to 0s? (Defaults: TRUE).
samples.in.rows	If TRUE, the samples are considered to be in the rows. Otherwise, it is assumed that they are in the columns. (Defaults: FALSE).
comp	If X is a list, this argument indicates how the ordinal/ranking variables are combined. Options are: "mean" and "sum". (Defaults: "mean").

Value

Kernel matrix (dimension: *NxN*).

References

Jiao, Y. and Vert, J.P. The Kendall and Mallows kernels for permutations. International Conference on Machine Learning. PMLR, 2015. Link

Examples

```
# 3 people are given a list of 10 colors. They rank them from most (1) to least
# (10) favorite
color_list <-  c("black","blue","green","grey","lightblue","orange","purple",
"red","white","yellow")
survey1 <- 1:10
survey2 <- 10:1
survey3 <- sample(10)
color <- cbind(survey1,survey2,survey3) # Samples in columns
rownames(color) <- color_list
Kendall(color)

# The same 3 people are asked the number of times they ate 5 different kinds of
# food during the last month:
```

```
food <- matrix(c(10, 1,18, 25,30, 7, 5,20, 5, 12, 7,20, 20, 3,22),ncol=5,nrow=3)
rownames(food) <- colnames(color)
colnames(food) <- c("spinach", "chicken", "beef" , "salad","lentils")
# (we can observe that, for person 2, vegetables << meat, while for person 3
# is the other way around)
Kendall(food,samples.in.rows=TRUE)

# We can combine this results:
dataset <- list(color=color,food=t(food)) #All samples in columns
Kendall(dataset)
```

kPCA *Kernel PCA*

Description

‘kPCA()‘ computes the kernel PCA from a kernel matrix and, if desired, produces a plot. The contribution of the original variables to the Principal Components (PCs), sometimes referred as "loadings", is NOT returned (to do so, go to ‘kPCA_imp()‘).

Usage

```
kPCA(
  K,
  center = TRUE,
  Ktest = NULL,
  plot = NULL,
  y = NULL,
  colors = "black",
  na_col = "grey70",
  title = "Kernel PCA",
  pos_leg = "right",
  name_leg = "",
  labels = NULL,
  ellipse = NULL
)
```

Arguments

`K` Kernel matrix (class "matrix").

`center` A logical value. If TRUE, the variables are zero-centered before the PCA. (Defaults: TRUE).

`Ktest` (optional) An additional kernel matrix corresponding to test samples, with dimension *Ntest x Ntraining*. These new samples are projected (using the color defined by ‘na_col‘) over the kernel PCA computed from K. Remember than the data that generated ‘Ktest‘ should be centered beforehand, using the same values used for centering ‘K‘.

`plot` (optional) A ‘ggplot2‘ is displayed. The input should be a vector of integers with length 2, corresponding to the two Principal Components to be displayed in the plot.

`y` (optional) A factor, or a numeric vector, with length equal to ‘nrow(K)‘ (number of samples). This parameter allows to paint the points with different colors.

`colors` A single color, or a vector of colors. If ‘y‘ is numeric, a gradient of colors between the first and the second entry will be used to paint the points. (Defaults: "black").

`na_col` Color of the entries that have a NA in the parameter ‘y‘, or the entries corresponding to ‘Ktest‘ (when ‘Ktest‘ is not NULL). Otherwise, this parameter is ignored.

`title` Plot title.

`pos_leg` Position of the legend.

`name_leg` Title of the legend. (Defaults: blank)

`labels` (optional) A vector of the same length than nrow(K). A name will be displayed next to each point.

`ellipse` (optional) A float between 0 and 1. An ellipse will be drawn for each group of points defined by ‘y‘. Here ‘y‘ should be of class "factor." This parameter will indicate the spread of the ellipse.

Details

As the ordinary PCA, kernel PCA can be used to summarize, visualize and/or create new features of a dataset. Data can be projected in a linear or nonlinear way, depending on the kernel used. When the kernel is ‘Linear()‘, kernel PCA is equivalent to ordinary PCA.

Value

A list with two objects:

* The PCA projection (class "matrix"). Please note that if K was computed from a *NxD* table with *N > D*, only the first *N-D* PCs may be useful.

* (optional) A ‘ggplot2‘ plot of the selected PCs.

Examples

```
dat <- matrix(rnorm(150),ncol=50,nrow=30)
K <- Linear(dat)

## Projection's coordinates only:
pca <- kPCA(K)

## Coordinates + plot of the two first principal components (PC1 and PC2):
pca <- kPCA(K,plot=1:2, colors = "coral2")
pca$plot
```

kPCA_arrows *Plot the original variables' contribution to a PCA plot*

Description

'kPCA_arrows()' draws arrows on a (kernel) PCA plot to represent the contribution of the original variables to the two displayed Principal Components (PCs).

Usage

```
kPCA_arrows(plot, contributions, colour = "steelblue", size = 4, ...)
```

Arguments

`plot`	A kernel PCA plot generated by 'kPCA()'.
`contributions`	The variables contributions, for instance obtained via 'kPCA_imp()'. It is not mandatory to draw all the original variables; a subset of interest can be passed on to this argument.
`colour`	Color of arrows and labels. (Defaults: "steelblue").
`size`	Size of the labels. (Defaults: 4).
`...`	Additional parameters passed on to geom_segments() and geom_text().

Details

It is important to note that the arrows are scaled to match the samples' projection plot. Thus, arrows' directions are correct, but do not expect that their magnitudes match the output of 'kPCA_imp()' or other functions('prcomp', 'princomp...'). (Nevertheless, they should at least be proportional to the real magnitudes.)

Value

The PCA plot with the arrows ('ggplot2' object).

Examples

```
dat <- matrix(rnorm(500),ncol=10,nrow=50)
K <- Linear(dat)

## Computing the kernel PCA. The plot represents PC1 and PC2:
kpca <- kPCA(K,plot=1:2)

## Computing the contributions to all the PCS:
pcs <- kPCA_imp(dat,secure=FALSE)

## We will draw the arrows for PC1 and PC2.
contributions <- t(pcs$loadings[1:2,])
rownames(contributions) <- 1:10
kPCA_arrows(plot=kpca$plot,contributions=contributions)
```

kPCA_imp *Contributions of the variables to the Principal Components ("loadings")*

Description

‘kPCA_imp()‘ performs a PCA and a kernel PCA simultaneously and returns the contributions of the variables to the Principal Components (sometimes, these contributions are called "loadings") in Feature Space. Optionally, it can also return the samples’ projection (cropped to the relevant PCs) and the values used to centering the variables in Feature Space. It does not return any plot, nor it projects test data. To do so, please use ‘kPCA()‘.

Usage

```
kPCA_imp(DATA, center = TRUE, projected = NULL, secure = FALSE)
```

Arguments

`DATA`	A matrix or data.frame (NOT a kernel matrix) containing the data in feature space. Please note that nrow(DATA) should be higher than ncol(DATA). If the Linear kernel is used, this feature space is simply the original space.
`center`	A logical value. If TRUE, the variables are zero-centered. (Defaults: TRUE).
`projected`	(optional) If desired, the PCA projection (generated, for example, by ‘kPCA()‘) can be included. If DATA is big (especially in the number of rows) this may save some computation time.
`secure`	(optional) If TRUE, it tests the quality of the loadings This may be slow. (Defaults: FALSE).

Details

This function may be not valid for all kernels. Do not use it with the RBF, Laplacian, Bray-Curtis, Jaccard/Ruzicka, or Kendall’s tau kernels unless you know exactly what you are doing.

Value

A list with three objects:

* The PCA projection (class "matrix") using only the relevant Principal Components.

* The loadings.

* The values used to center each variable in Feature Space.

Examples

```
dat <- matrix(rnorm(150),ncol=30,nrow=50)
contributions <- kPCA_imp(dat)
contributions$loadings[c("PC1","PC2"),1:5]
```

KTA *Kernel-target alignment*

Description

‘KTA()‘ computes the alignment between a kernel matrix and a target variable.

Usage

```
KTA(K, y)
```

Arguments

K	A kernel matrix (class: "matrix").
y	The target variable. A factor with two levels.

Value

Alignment value.

Examples

```
K1 <- RBF(iris[1:100,1:4],g=0.1)
y <- factor(iris[1:100,5])
KTA(K1,y)
```

Laplace *Laplacian kernel*

Description

‘Laplace()‘ computes the laplacian kernel between all possible pairs of rows of a matrix or data.frame with dimension *NxD*.

Usage

```
Laplace(X, g = NULL)
```

Arguments

X	Matrix or data.frame that contains real numbers ("integer", "float" or "double").
g	Gamma hyperparameter. If g=0 or NULL, ‘Laplace()‘ returns the Manhattan distance (L1 norm between two vectors). (Defaults=NULL)

Details

Let x_i, x_j be two real vectors. Then, the laplacian kernel is defined as:

$$K_{Lapl}(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|_1)$$

Value

Kernel matrix (dimension: *NxN*).

Examples

```
dat <- matrix(rnorm(250),ncol=50,nrow=5)
Laplace(dat,g=0.1)
```

Linear *Linear kernel*

Description

'Linear()' computes the inner product between all possible pairs of rows of a matrix or data.frame with dimension *NxD*.

Usage

```
Linear(X, cos.norm = FALSE, coeff = NULL)
```

Arguments

X	Matrix or data.frame that contains real numbers ("integer", "float" or "double").
cos.norm	Should the resulting kernel matrix be cosine normalized? (Defaults: FALSE).
coeff	(optional) A vector of length *D* that weights each one of the features (columns). When cos.norm=TRUE, 'Linear()' first does the weighting and then the cosine-normalization.

Value

Kernel matrix (dimension: *NxN*).

Examples

```
dat <- matrix(rnorm(250),ncol=50,nrow=5)
Linear(dat)
```

minmax *Minmax normalization*

Description

Minmax normalization. Custom min/max values may be passed to the function.

Usage

```
minmax(X, rows = FALSE, values = NULL)
```

Arguments

X Numeric matrix or data.frame of any size.

rows If TRUE, the minmax normalization is done by row; otherwise, it is done by column. (Defaults: FALSE)

values (optional) A list containing two elements, the "max" values and the "min" values. If no value is passed, the typical minmax normalization (which normalizes the dataset between 0 and 1) is computed with the observed maximum and minimum value in each column (or row) of X.

Value

Minmax-normalized X.

Examples

```
dat <- matrix(rnorm(100),ncol=10,nrow=10)
dat_minmax <- minmax(dat)
apply(dat_minmax,2,min) ## Min values = 0
apply(dat_minmax,2,max) ## Max values = 1
# We can also explicitly state the max and min values:
values <- list(min=apply(dat,2,min),max=apply(dat,2,max))
dat_minmax <- minmax(dat,values=values)
```

MKC *Multiple Kernel (Matrices) Combination*

Description

Combination of kernel matrices coming from different datasets / feature types into a single kernel matrix.

Usage

```
MKC(K, coeff = NULL)
```

Arguments

`K` A three-dimensional *NxDxM* array containing *M* kernel matrices.

`coeff` A vector of length *M* with the weight of each kernel matrix. If NULL, all kernel matrices have the same weight. (Defaults: NULL)

Value

A kernel matrix.

Examples

```
# For illustrating a possible use of this function, we work with a dataset
# that contains numeric and categorical features.

summary(mtcars)
cat_feat_idx <- which(colnames(mtcars) %in% c(”vs”, ”am”))

# vs and am are categorical variables. We make a list, with the numeric features
# in the first element and the categorical features in the second:
DATA <- list(num=mtcars[,-cat_feat_idx], cat=mtcars[,cat_feat_idx])
# Our N, D and M dimensions are:
N <- nrow(mtcars); D <- ncol(mtcars); M <- length(DATA)

# Now we prepare a kernel matrix:
K <- array(dim=c(N,N,M))
K[,,1] <- Linear(DATA[[1]],cos.norm = TRUE) ## Kernel for numeric data
K[,,2] <- Dirac(DATA[[2]]) ## Kernel for categorical data

# Here, K1 has the same weight than K2 when computing the final kernel, although
# K1 has 9 variables and K2 has only 2.
Kconsensus <- MKC(K)
Kconsensus[1:5,1:5]

# If we want to weight equally each one of the 11 variables in the final
# kernel, K1 will weight 9/11 and K2 2/11.
coeff <- sapply(DATA,ncol)
coeff
Kweighted <- MKC(K,coeff=coeff)
Kweighted[1:5,1:5]
```

`nmse` *NMSE (Normalized Mean Squared Error)*

Description

‘nmse()‘ computes the Normalized Mean Squared Error between the output of a regression model and the actual values of the target.

Usage

```
nmse(target, pred)
```

Arguments

`target`	Numeric vector containing the actual values.
`pred`	Numeric vector containing the predicted values. (The order should be the same than in the target)

Details

The Normalized Mean Squared error is defined as:

$$NMSE = MSE/((N-1)*var(target))$$

where MSE is the Mean Squared Error.

Value

The normalized mean squared error (a single value).

Examples

```
y <- 1:10
y_pred <- y+rnorm(10)
nmse(y,y_pred)
```

Normal_CI *Confidence Interval using Normal Approximation*

Description

'Normal_CI()' computes the Confidence Interval (CI) of a performance measure (for instance, accuracy) using normal approximation. Thus, it is advisable that the test has a size of, at least, 30 instances.

Usage

```
Normal_CI(value, ntest, confidence = 95)
```

Arguments

`value`	Performance value (a single value).
`ntest`	Test set size (a single value).
`confidence`	Confidence level; for instance, 95% or 99%. (Defaults: 95).

Value

A vector containing the CI.

Examples

```
# Computing accuracy
y <- c(rep("a",30),rep("b",20))
y_pred <- c(rep("a",20),rep("b",30))
ct <- table(y,y_pred)
accuracy <- Acc(ct)
# Computing 95%CI
Normal_CI(accuracy, ntest=length(y), confidence=95)
```

`plotImp` *Importance barplot*

Description

‘plotImp()‘ displays the barplot of a numeric vector, which is assumed to contain the features importance (from a prediction model) or the contribution of each original variable to a Principal Component (PCA). In the barplot, features/PCs are sorted by decreasing importance.

Usage

```
plotImp(
  x,
  y = NULL,
  relative = TRUE,
  absolute = TRUE,
  nfeat = NULL,
  names = NULL,
  main = NULL,
  xlim = NULL,
  color = "grey",
  leftmargin = NULL,
  ylegend = NULL,
  leg_pos = "right",
  ...
)
```

Arguments

`x` Numeric vector containing the importances.

`y` (optional) Numeric vector containing a different, independent variable to be contrasted with the feature importances. Should have the same length and order than ‘x‘.

`relative` If TRUE, the barplot will display relative importances. (Defaults: TRUE).

`absolute` If FALSE, the bars may be positive or negative, which will affect the order of the features Otherwise, the absolute value of ‘x‘ will be taken (Defaults: TRUE).

`nfeat` (optional) The number of top (most important) features displayed in the plot.

`names` (optional) The names of the features, in the same order than ‘x‘.

`main` (optional) Plot title.

`xlim` (optional) A numeric vector. If absent, the minimum and maximum value of ‘x‘ will be used to establish the axis’ range.

`color` Color(s) chosen for the bars. A single value or a vector. (Defaults: "grey").

`leftmargin` (optional) Left margin space for the plot.

`ylegend` (optional) It allows to add a text explaining what is ‘y‘ (only if ‘y‘ is not NULL).

`leg_pos` If ‘ylegend‘ is TRUE, the position of the legend. (Defaults: "right").

`...` (optional) Additional arguments (such as ‘axes‘, ‘asp‘,...) and graphical parameters (such as ‘par‘). See ‘?graphics::barplot()‘.

Value

A list containing:

* The vector of importances in decreasing order. When ‘nfeat‘ is not NULL, only the top ‘nfeat‘ are returned.

* The cumulative sum of (absolute) importances.

* A numeric vector giving the coordinates of all the drawn bars’ midpoints.

Examples

```
importances <- rnorm(30)
names_imp <- paste0("Feat",1:length(importances))

plot1 <- plotImp(x=importances,names=names_imp,main="Barplot")
plot2 <- plotImp(x=importances,names=names_imp,relative=FALSE,
main="Barplot",nfeat=10)
plot3 <- plotImp(x=importances,names=names_imp,absolute=FALSE,
main="Barplot",color="coral2")
```

`Prec` *Precision or PPV*

Description

‘Prec()‘ computes the Precision of PPV (Positive Predictive Value) between the output of a classification model and the actual values of the target. The precision of each class can be aggregated. Macro-precision is the average of the precision of each classes. Micro-precision is the weighted average.

Usage

```
Prec(ct, multi.class = "macro")
```

Arguments

`ct`	Confusion Matrix.
`multi.class`	Should the results of each class be aggregated, and how? Options: "none", "macro", "micro". (Defaults: "macro").

Value

PPV (a single value).

Examples

```
y <- c(rep("a",3),rep("b",2))
y_pred <- c(rep("a",2),rep("b",3))
ct <- table(y,y_pred)
Prec(ct)
```

`Procrustes` *Procrustes Analysis*

Description

Procrustes Analysis compares two PCA/PCoA/MDS/other ordination methods' projections after "removing" the translation, scaling and rotation effects. Thus, they are compared in their configuration of "maximum similarity". Samples in the two projections should be related. The similarity of the projections X1 and X2 is quantified using a correlation-like statistic derived from the symmetric Procrustes sum of squared differences between X1 and X2.

Usage

```
Procrustes(X1, X2, plot = NULL, labels = NULL)
```

Arguments

`X1`	A matrix or data.frame containing a PCA/PCoA/MDS projection.
`X2`	A second matrix or data.frame containing a different PCA/PCoA/MDS projection, with the same number of rows than X1.
`plot`	(optional) A ‘ggplot2‘ is displayed. The input should be a vector of integers with length 2, corresponding to the two Principal Components to be displayed in the plot.
`labels`	(optional) A vector of the same length than nrow(X1), or instead, nrow(X1)+nrow(X2). A name will be displayed next to each point.

Details

‘Procrustes()‘ performs a Procrustes Analysis equivalent to ‘vegan::procrustes(X,Y,scale=FALSE,symmetric=TRUE)‘.

Value

A list containing:

* X1 (zero-centered and scaled).

* X2 superimposed over X1 (after translating, scaling and rotating X2).

* Procrustes correlation between X1 and X2.

* (optional) A ‘ggplot2‘ plot.

Examples

```
data1 <- matrix(rnorm(900),ncol=30,nrow=30)
data2 <- matrix(rnorm(900),ncol=30,nrow=30)
pca1 <- kPCA(Linear(data1),center=TRUE)
pca2 <- kPCA(Linear(data2),center=TRUE)
procr <- Procrustes(pca1,pca2)
# Procrustean correlation between pca1 and pca2:
procr$pro.cor
# With plot (first two axes):
procr <- Procrustes(pca1,pca2,plot=1:2,labels=1:30)
procr$plot
```

RBF *Gaussian RBF (Radial Basis Function) kernel*

Description

‘RBF()‘ computes the RBF kernel between all possible pairs of rows of a matrix or data.frame with dimension *NxD*.

Usage

```
RBF(X, g = NULL)
```

Arguments

`X` Matrix or data.frame that contains real numbers ("integer", "float" or "double").

`g` Gamma hyperparameter. If g=0 or NULL, ‘RBF()‘ returns the matrix of squared Euclidean distances instead of the RBF kernel matrix. (Defaults=NULL).

Details

Let x_i, x_j be two real vectors. Then, the RBF kernel is defined as:

$$K_{RBF}(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

Sometimes the RBF kernel is given a hyperparameter called sigma. In that case: $\gamma = 1/\sigma^2$.

Value

Kernel matrix (dimension: *NxN*).

Examples

```
dat <- matrix(rnorm(250),ncol=50,nrow=5)
RBF(dat,g=0.1)
```

Rec *Recall or Sensitivity or TPR*

Description

‘Rec()‘ computes the Recall, also known as Sensitivity or TPR (True Positive Rate), between the output of a classification model and the actual values of the target.

Usage

```
Rec(ct, multi.class = "macro")
```

Arguments

ct	Confusion Matrix.
multi.class	Should the results of each class be aggregated, and how? Options: "none", "macro", "micro". (Defaults: "macro").

Value

TPR (a single value).

Examples

```
y <- c(rep("a",3),rep("b",2))
y_pred <- c(rep("a",2),rep("b",3))
ct <- table(y,y_pred)
Rec(ct)
```

`showdata` *Showdata*

Description

A toy dataset that contains the results of a (fictional) survey commissioned from a well-known streaming platform. The platform invited 100 people to watch footage of their new show before the premiere. After that, the participants were asked to pick their favorite color, actress, actors and shows from a list. Finally, they were asked to disclose if they liked the new show.

Usage

```
showdata
```

Format

A data.frame with 100 rows and 5 factor variables:

Favorite.color Favorite color

Favorite.actress Favorite actress

Favorite.actor Favorite actor

Favorite.show Favorite show

Liked.new.show Do you like the new show?

Source

Own

`simK` *Kernel matrix similarity*

Description

‘simK()‘ computes the similarity between kernel matrices.

Usage

```
simK(Klist)
```

Arguments

`Klist` A list of *M* kernel matrices with identical *NxN* dimension.

Details

It is a wrapper of ‘Frobenius()‘.

Value

Kernel matrix (dimension: *MxM*).

Examples

```
K1 <- Linear(matrix(rnorm(7500),ncol=150,nrow=50))
K2 <- Linear(matrix(rnorm(7500),ncol=150,nrow=50))
K3 <- Linear(matrix(rnorm(7500),ncol=150,nrow=50))

simK(list(K1,K2,K3))
```

soil *Soil microbiota (raw counts)*

Description

Bacterial abundances in 89 soils from across North and South America.

Usage

```
soil
```

Format

A list containing the following elements:

abund Bacterial abundances of 7396 taxa in 88 sites.

metadata Samples' metadata

taxonomy Taxonomic information

References

Lauber CL, Hamady M, Knight R, Fierer N. Pyrosequencing-based assessment of soil pH as a predictor of soil bacterial community structure at the continental scale. Appl Environ Microbiol. 2009 Aug;75(15):5111-20. doi: 10.1128/AEM.00335-09.

Spe	*Specificity or TNR*

Description

‘Spe()‘ computes the Specificity or TNR (True Negative Rate) between the output of a classification prediction model and the actual values of the target.

Usage

```
Spe(ct, multi.class = "macro")
```

Arguments

ct	Confusion Matrix.
multi.class	Should the results of each class be aggregated, and how? Options: "none", "macro", "micro". (Defaults: "macro").

Value

TNR (a single value).

Examples

```
y <- c(rep("a",3),rep("b",2))
y_pred <- c(rep("a",2),rep("b",3))
ct <- table(y,y_pred)
Spe(ct)
```

Spectrum	*Spectrum kernel*

Description

‘Spectrum()‘ computes the basic Spectrum kernel between strings. This kernel computes the similarity of two strings by counting how many matching substrings of length l are present in each one.

Usage

```
Spectrum(
  x,
  alphabet,
  l = 1,
  group.ids = NULL,
  weights = NULL,
  feat_space = FALSE,
  cos.norm = FALSE
)
```

Arguments

`x` Vector of strings (length *N*).

`alphabet` Alphabet of reference.

`l` Length of the substrings.

`group.ids` (optional) A vector with ids. It allows to compute the kernel over groups of strings within x, instead of the individual strings.

`weights` (optional) A numeric vector as long as x. It allows to weight differently each one of the strings.

`feat_space` If FALSE, only the kernel matrix is returned. Otherwise, the feature space (i.e. a table with the number of times that a substring of length *l* appears in each string) is also returned (Defaults: FALSE).

`cos.norm` Should the resulting kernel matrix be cosine normalized? (Defaults: FALSE).

Details

In large datasets this function may be slow. In that case, you may use the ‘stringdot()‘ function of the ‘kernlab‘ package, or the ‘spectrumKernel()‘ function of the ‘kebabs‘ package.

Value

Kernel matrix (dimension: *NxN*), or a list with the kernel matrix and the feature space.

References

Leslie, C., Eskin, E., and Noble, W.S. The spectrum kernel: a string kernel for SVM protein classification. Pac Symp Biocomput. 2002:564-75. PMID: 11928508. Link

Examples

```
## Examples of alphabets. _ stands for a blank space, a gap, or the
## start or the end of sequence)
NT <- c("A","C","G","T","_") # DNA nucleotides
AA <- c("A","C","D","E","F","G","H","I","K","L","M","N","P","Q","R","S","T",
"V","W","Y","_") ##canonical aminoacids
letters_ <- c(letters,"_")
## Example of data
```

```
strings <- c("hello_world","hello_word","hola_mon","kaixo_mundua",
"saluton_mondo","ola_mundo", "bonjour_le_monde")
names(strings) <- c("english1","english_typo","catalan","basque",
"esperanto","galician","french")
## Computing the kernel:
Spectrum(strings,alphabet=letters_,l=2)
```

svm_imp *SVM feature importance*

Description

Recovering the features importances from a SVM model.

Usage

```
svm_imp(
  X,
  svindx,
  coeff,
  result = "absolute",
  cos.norm = FALSE,
  center = FALSE,
  scale = FALSE
)
```

Arguments

X	Matrix or data.frame that contains real numbers ("integer", "float" or "double"). X is NOT the kernel matrix, but the original dataset used to compute the kernel matrix.
svindx	Indices of the support vectors.
coeff	target * alpha.
result	A string. If "absolute", the absolute values of the importances are returned. If "squared", the squared values are returned. Any other input will result in the original (positive and/or negative) importance values (see Details). (Defaults: "absolute").
cos.norm	Boolean. Was the data cosine normalized prior to training the model? (Defaults: FALSE).
center	Boolean. Was the data centered prior to training the model? (Defaults: FALSE).
scale	Boolean. Was the data scaled prior to training the model? (Defaults: FALSE).

Details

This function may be not valid for all kernels. Do not use it with the RBF, Laplacian, Bray-Curtis, Jaccard/Ruzicka, or Kendall's tau kernels unless you know exactly what you are doing.

Usually the sign of the importances is irrelevant, thus justifying working with the absolute or squared values; see for instance Guyon et al. (2002). Some classification tasks are an exception to this, when it can be demonstrated that the feature space is strictly nonnegative. In that case, a positive importance implies that a feature contributes to the "positive" class, and the same with a negative importance and the "negative" class.

Value

The importance of each feature (a vector).

References

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002) Gene selection for cancer classification using support vector machines. Machine learning, 46, 389-422. Link

Examples

```
data1 <- iris[1:100,]
sv_index <- c( 24, 42, 58, 99)
coefficients <- c(-0.2670988, -0.3582848,  0.2129282,  0.4124554)
# This SV and coefficients were obtained from a model generated with kernlab:
# model <- kernlab::ksvm(Species ~ .,data=data1, kernel="vanilladot",scaled = TRUE)
# sv_index <- unlist(kernlab::alphaindex(model))
# coefficients <- kernlab::unlist(coef(model))
# Now we compute the importances:
svm_imp(X=data1[,-5],svindx=sv_index,coeff=coefficients,center=TRUE,scale=TRUE)
```

`TSS` *Total Sum Scaling*

Description

This function transforms a dataset from absolute to relative frequencies (by row or column).

Usage

```
TSS(X, rows = TRUE)
```

Arguments

`X`	Numeric matrix or data.frame of any size containing absolute frequencies.
`rows`	If TRUE, the operation is done by row; otherwise, it is done by column. (Defaults: TRUE).

Value

A relative frequency matrix or data.frame with the same dimension than X.

Examples

```
dat <- matrix(rnorm(50),ncol=5,nrow=10)
TSS(dat) #It can be checked that, after scaling, the sum of each row is equal to 1.
```

---

vonNeumann *Von Neumann entropy*

Description

‘vonNeumann()‘ computes the von Neumann entropy of a kernel matrix. Entropy values close to 0 indicate that all its elements are very similar, which may result in underfitting when training a prediction model. Instead, values close to 1 indicate a high variability which may produce overfitting.

Usage

```
vonNeumann(K)
```

Arguments

K	Kernel matrix (class "matrix").

Value

Von Neumann entropy (a single value).

References

Belanche-Muñoz, L.A. and Wiejacha, M. (2023) Analysis of Kernel Matrices via the von Neumann Entropy and Its Relation to RVM Performances. Entropy, 25, 154. doi:10.3390/e25010154. Link

Examples

```
data <- matrix(rnorm(150),ncol=50,nrow=30)
K <- Linear(data)
vonNeumann(K)
```

Index