Package ‘mixKernel’

February 26, 2020

Type Package

Title Omics Data Integration Using Kernel Methods

Version 0.4

Date 2020-02-20

Depends R (>= 3.5.0), mixOmics, ggplot2, reticulate (>= 1.14)

Imports vegan, phyloseq, corrplot, psych, quadprog, LDRTools, Matrix, methods

Maintainer Jerome Mariette <jerome.mariette@inrae.fr>

Description Kernel-based methods are powerful methods for integrating heterogeneous types of data. mixKernel aims at providing methods to combine kernel for unsupervised exploratory analysis. Different solutions are provided to compute a meta-kernel, in a consensus way or in a way that best preserves the original topology of the data. mixKernel also integrates kernel PCA to visualize similarities between samples in a non linear space and from the multiple source point of view. Functions to assess and display important variables are also provided in the package. Jerome Mariette and Nathalie Villa-Vialaneix (2017) <doi:10.1093/bioinformatics/btx682>.

License GPL (>= 2)

Repository CRAN

Date/Publication 2020-02-26 12:50:06 UTC

NeedsCompilation no

LazyData true

Config/reticulate list(packages = list(list(package = ``autograd'', pip = TRUE), list(package = ``numpy'', pip = TRUE), list(package = ``scipy'', pip = TRUE)))

Author Jerome Mariette [aut, cre],
Celine Brouard [aut],
Remi Flamary [aut],
Nathalie Vialaneix [aut]

R topics documented:

cim.kernel 2
combine.kernels 3
compute.kernel 5
kernel.pca 7
kernel.pca.permute 8
mixKernel.users.guide 9
plotVar.kernel.pca 10
TARAoceans 11
ukfs 12

Index **14**

`cim.kernel` *Compute and display similarities between multiple kernels*

Description

Compute cosine from Frobenius norm between kernels and display the corresponding correlation plot.

Usage

```
cim.kernel(..., scale = TRUE,
           method = c("circle", "square", "number", "shade", "color", "pie"))
```

Arguments

`...`	list of kernels (called 'blocks') computed on different datasets and measured on the same samples.
`scale`	boleean. If `scale = TRUE`, each block is standardized to zero mean and unit variance and cosine normalization is performed on the kernel. Default: `TRUE`.
`method`	character. The visualization method to be used. Currently, seven methods are supported (see Details).

Details

The displayed similarities are the kernel generalization of the RV-coefficient described in Lavit *et al.*, 1994.

The plot is displayed using the `corrplot` package. Seven visualization methods are implemented: `"circle"` (default), `"square"`, `"number"`, `"pie"`, `"shade"` and `"color"`. Circle and square areas are proportional to the absolute value of corresponding similarities coefficients.

Value

`cim.kernel` returns a matrix containing the cosine from Frobenius norm between kernels.

Author(s)

Jerome Mariette <jerome.mariette@inrae.fr>

Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

References

Lavit C., Escoufier Y., Sabatier R. and Traissac P. (1994). The ACT (STATIS method). *Computational Statistics and Data Analysis*, **18**(1), 97-119.

Mariette J. and Villa-Vialaneix N. (2017). Integrating *TARA* Oceans datasets using unsupervised multiple kernel learning. *Preprint*

See Also

`compute.kernel`

Examples

```
data(TARAoceans)

# compute one kernel per dataset
phychem.kernel <- compute.kernel(TARAoceans$phychem, kernel.func = "linear")
pro.phylo.kernel <- compute.kernel(TARAoceans$pro.phylo, kernel.func = "abundance")
pro.NOGs.kernel <- compute.kernel(TARAoceans$pro.NOGs, kernel.func = "abundance")

# display similarities between kernels
cim.kernel(phychem = phychem.kernel,
           pro.phylo = pro.phylo.kernel,
           pro.NOGs = pro.NOGs.kernel,
           method = "square")
```

`combine.kernels` *Combine multiple kernels into a meta-kernel*

Description

Compute multiple kernels into a single meta-kernel

Usage

```
combine.kernels(..., scale = TRUE,
                method = c("full-UMKL", "STATIS-UMKL", "sparse-UMKL"), knn = 5,
                rho = 20)
```

Arguments

`...`	list of kernels (called 'blocks') computed on different datasets and measured on the same samples.
`scale`	boleean. If `scale = TRUE`, each block is standardized to zero mean and unit variance and cosine normalization is performed on the kernel. Default: `TRUE`.
`method`	character. Which method should be used to compute the meta-kernel. Default: `"full-UMKL"`.
`knn`	integer. If `method = "sparse-UMKL"` or `method = "full-UMKL"`, number of neighbors used to get a proxy of the local topology of the datasets from each kernel. Default: `5`.
`rho`	integer. Parameters for the augmented Lagrangian method. Default: `20`.

Details

The arguments `method` allows to specify the Unsupervised Multiple Kernel Learning (UMKL) method to use:

- `"STATIS-UMKL"`: combines input kernels into the best consensus of all kernels;
- `"full-UMKL"`: computes a kernel that minimizes the distortion between the meta-kernel and the k-NN graphs obtained from all input kernels;
- `"sparse-UMKL"`: a sparse variant of the `"full-UMKL"` approach.

Value

`combine.kernels` returns an object of classes `"kernel"` and `"metaKernel"`, a list that contains the following components:

- kernel: the computed meta-kernel matrix;
- X: the dataset from which the kernel has been computed, as given by the function `compute.kernel`. Can be `NULL` if a kernel matrix was passed to this function;
- weights: a vector containing the weights used to combine the kernels.

References

Mariette J. and Villa-Vialaneix N. (2017). Integrating *TARA* Oceans datasets using unsupervised multiple kernel learning. *Preprint*

Author(s)

Jerome Mariette <jerome.mariette@inrae.fr>

Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

See Also

`compute.kernel`, `kernel.pca`

Examples

```
data(TARAoceans)

# compute one kernel per dataset
phychem.kernel <- compute.kernel(TARAoceans$phychem, kernel.func = "linear")
pro.phylo.kernel <- compute.kernel(TARAoceans$pro.phylo, kernel.func = "abundance")
pro.NOGs.kernel <- compute.kernel(TARAoceans$pro.NOGs, kernel.func = "abundance")

# compute the meta kernel
meta.kernel <- combine.kernels(phychem = phychem.kernel,
                               pro.phylo = pro.phylo.kernel,
                               pro.NOGs = pro.NOGs.kernel,
                               method = "full-UMKL")
```

`compute.kernel` *Compute a kernel*

Description

Compute a kernel from a given data matrix.

Usage

```
compute.kernel(X, kernel.func = "linear", ..., test.pos.semidef = FALSE)
```

Arguments

`X`	a numeric matrix (or data frame) used to compute the kernel. `NA`s not allowed.
`kernel.func`	the kernel function to use. This parameter can be set to any user defined kernel function. Widely used kernel functions are pre-implemented, that can be used by setting `kernel.func` to one of the following strings: `"kidentity"`, `"abundance"`, `"linear"`, `"gaussian.radial.basis"`, `"poisson"` or `"phylogenetic"`. Default: `"linear"`.
`...`	the kernel function arguments. Valid parameters for pre-implemented kernels are: • `phylogenetic.tree` (`"phylogenetic"`): an instance of phylo-class that contains a phylogenetic tree (required). • `scale` (`"linear"` or `"gaussian.radial.basis"`): logical. Should the variables be scaled to unit variance prior the kernel computation? Default: `TRUE`. • `sigma` (`"gaussian.radial.basis"`): double. The inverse kernel width used by `"gaussian.radial.basis"`. • `method` (`"phylogenetic"` or `"abundance"`): character. Can be `"unifrac"` or `"wunifrac"` for `"phylogenetic"`. Which dissimilarity to use for `"abundance"`: one of `"bray"`, `"euclidean"`, `"canberra"`, `"manhattan"`, `"kulczynski"`, `"jaccard"`, `"gower"`, `"altGower"`, `"morisita"`, `"horn"`, `"mountford"`, `"raup"`, `"binomial"`, `"chao"` and `"cao"`.

- `normalization` (`"poisson"`): character. Can be `"deseq"` (more robust), `"mle"` (less robust) or `"quantile"`.

`test.pos.semidef`
: boleean. If `test.pos.semidef = TRUE`, the resulting matrix is tested to be positive-semidefinite.

Value

`compute.kernel` returns an object of classes `"kernel"`, a list that contains the following components:

`kernel`	: the computed kernel matrix.
`X`	: the original dataset. If `"kidentity"`, `X` is set to `NULL`.
`kernel.func`	: the kernel function used.
`kernel.args`	: the arguments used to compute the kernel.

Author(s)

Jerome Mariette <jerome.mariette@inrae.fr>

Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

References

Lozupone C. and Knight R. (2005). UniFrac: a new phylogenetic method for comparing microbial communities. *Applied and Environmental Microbiology*, **71**(12), 8228-8235.

Lozupone C., Hamady M., Kelley S.T. and Knight R. (2007). Quantitative and qualitative beta diversity measures lead to different insights into factors that structure microbial communities. *Applied and Environmental Microbiology*, **73**(5), 1576-1585.

Witten D. (2011). Classification and clustering of sequencing data using a Poisson model. *Annals of Applied Statistics*, **5**(4), 2493-2518.

See Also

`combine.kernels`, `kernel.pca`

Examples

```
data(TARAoceans)
pro.NOGs.kernel <- compute.kernel(TARAoceans$pro.NOGs, kernel.func = "abundance")
```

`kernel.pca` *Kernel Principal Components Analysis*

Description

Performs a kernel PCA.

Usage

```
kernel.pca(K, ncomp = nrow(K$kernel))
```

Arguments

`K`	a kernel object obtained using either `compute.kernel` or `combine.kernels`.
`ncomp`	integer. Indicates the number of components to return.

Value

`kernel.pca` returns an object of classes `"kernel.pca"` and `"pca"`, which is a list containing the following entries:

- ncomp: the number of principal components;
- X: the input kernel matrix;
- kernel: the input kernel object provided by the user;
- sdev: the singular values (square root of the eigenvalues);
- rotation: the matrix of variable loadings (*i.e.*, a matrix whose columns contain the eigenvectors);
- loadings: same as ’rotation’ to keep the mixOmics spirit;
- x: same as ’rotation’ to keep the mixOmics spirit;

References

Scholkopf B., Smola A. and Muller K.R. (1998) Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, **10**, 1299-1319.

Author(s)

Jerome Mariette <jerome.mariette@inrae.fr>

Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

See Also

`compute.kernel`, `combine.kernels`

Examples

```
data(TARAoceans)
phychem.kernel <- compute.kernel(TARAoceans$phychem, kernel.func = "linear")
kernel.pca.result <- kernel.pca(phychem.kernel, ncomp = 3)
```

`kernel.pca.permute` *Assess variable importance*

Description

Assess importance of variables on a given PC component by computing the Crone-Crosby distance between original sample positions and sample positions obtain by a random permutation of the variables.

Usage

```
kernel.pca.permute(kpca.result, ncomp = 1, ..., directory = NULL)
```

Arguments

`kpca.result`	a kernel.pca object returned by the `kernel.pca` function.
`ncomp`	integer. Number of KPCA components used to compute the importance. Default: `1`.
`...`	list of character vectors. The parameter name must be the kernel name to be considered for permutation of variables. Provided vectors length has to be equal to the number of variables of the input dataset. A kernel is performed on each unique variables values. Crone-Crosby distances are computed on each KPCA performed on resulted kernels or meta-kernels and can be displayed using the `plotVar.kernel.pca`.
`directory`	character. To limit computational burden, this argument allows to store / read temporary computed kernels.

Value

`kernel.pca.permute` returns a copy of the input `kpca.result` results and add values in the three entries: `cc.distances`, `cc.variables` and `cc.blocks`.

References

Mariette J. and Villa-Vialaneix N. (2017). Integrating *TARA* Oceans datasets using unsupervised multiple kernel learning. *Preprint*

Crone L. and Crosby D. (1995). Statistical applications of a metric on subspaces to satellite meteorology. *Technometrics*, **37**(3), 324-328.

Author(s)

Jerome Mariette <jerome.mariette@inrae.fr>

Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

See Also

`compute.kernel`, `kernel.pca`, `plotVar.kernel.pca`

Examples

```
data(TARAoceans)

# compute one kernel for the psychem dataset
phychem.kernel <- compute.kernel(TARAoceans$phychem, kernel.func = "linear")
# perform a KPCA
kernel.pca.result <- kernel.pca(phychem.kernel)

# compute importance for all variables in this kernel
kernel.pca.result <- kernel.pca.permute(kernel.pca.result, phychem = colnames(TARAoceans$phychem))
```

`mixKernel.users.guide` *View mixKernel User's Guide*

Description

Find the location of the mixKernel User's Guide and optionnaly opens it

Usage

```
mixKernel.users.guide(html = TRUE, view = html)
```

Arguments

`html`	logical. Should the document returned by the function be the compiled PDF or the Rmd source. Default to `TRUE`
`view`	logical. Should the document be opened using the default HTML viewer? Default to `html`. It has no effect if `html = FALSE`

Details

If the operating system is not Windows, then the HTML viewer used is that given by `Sys.getenv("R_BROWSER")`. The HTML viewer can be changed using `Sys.setenv(R_BROWSER = )`.

Value

Character string giving the file location. If `html = TRUE` and `view = TRUE`, the HTML document reader is started and the User's Guide is opened in it.

Author(s)

Jerome Mariette <jerome.mariette@inrae.fr>

Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

Examples

```
mixKernel.users.guide(view = FALSE)
mixKernel.users.guide(html = FALSE)
## Not run: mixKernel.users.guide()
```

plotVar.kernel.pca *Plot importance of variables in kernel PCA*

Description

Provides a representation of variable importance in kernel PCA.

Usage

```
plotVar.kernel.pca(object, blocks = unique(object$cc.blocks), ndisplay = 5, ncol = 2, ...)
```

Arguments

`object`	: a kernel.pca object returned by `kernel.pca`.
`blocks`	a numerical vector indicating the block variables to display.
`ndisplay`	integer. The number of important variables per blocks shown in the representation. Default: `5`.
`ncol`	integer. Each block of variables is displayed in a separate subfigure. `ncol` sets the number of columns for the global figure. Default: `2`.
`...`	external arguments.

Details

`plotVar.kernel.pca` produces a barplot for each block. The variables for which the importance has been computed with `kernel.pca.permute` are displayed. The representation is limited to the `ndisplay` most important variables.

References

Crone L. and Crosby D. (1995). Statistical applications of a metric on subspaces to satellite meteorology. *Technometrics*, **37**(3), 324-328.

Author(s)

Jerome Mariette <jerome.mariette@inrae.fr>

Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

See Also

kernel.pca, kernel.pca.permute

Examples

```
data(TARAoceans)

# compute one kernel for the psychem dataset
phychem.kernel <- compute.kernel(TARAoceans$phychem, kernel.func = "linear")
# perform a KPCA
kernel.pca.result <- kernel.pca(phychem.kernel)
# compute importance for all variables in this kernel
kernel.pca.result <- kernel.pca.permute(kernel.pca.result, phychem = colnames(TARAoceans$phychem))

## Not run: plotVar.kernel.pca(kernel.pca.result, ndisplay = 10)
```

`TARAoceans` *TARA ocean microbiome data*

Description

The TARA Oceans expedition facilitated the study of plankton communities by providing oceans metagenomic data combined with environmental measures to the scientific community. This dataset focuses on 139 prokaryotic-enriched samples collected from 68 stations and spread across three depth layers: the surface (SRF), the deep chlorophyll maximum (DCM) layer and the mesopelagic (MES) zones. Samples were located in height different oceans or seas: Indian Ocean (IO), Mediterranean Sea (MS), North Atlantic Ocean (NAO), North Pacific Ocean (NPO), Red Sea (RS), South Atlantic Ocean (SAO), South Pacific Ocean (SPO) and South Ocean (SO). Here, only a subset of the original data is provided (1% of the 35,650 prokaryotic operational taxonomic units (OTUs) and of the 39,246 bacterial genes (NOGs) (selected at random).

Usage

```
data(TARAoceans)
```

Format

A list containing the following components:

`phychem` data matrix with 139 rows and 22 columns. Each row represents a sample and each column an environmental variable.

`pro.phylo` data matrix with 139 rows (samples) and 356 columns (prokaryotic OTUs).

`taxonomy` data matrix with 356 rows (prokaryotic OTUs) and 6 columns indicating the taxonomy of each OTU.

`phylogenetic.tree` a phylo object (see package 'ape') representing the prokaryotic OTUs phylogenetic tree.

`pro.NOGs` data matrix with 139 rows (samples) and 638 columns (NOGs).

`sample` a list containing three following entries (all three are character vectors): `name` (sample name), `ocean` (oceanic region of the sample) and `depth` (sample depth).

References

Sunagawa S., Coelho L.P., Chaffron S., Kultima J.R., Labadie K., Salazar F., Djahanschiri B., Zeller G., Mende D.R., Alberti A., Cornejo-Castillo F., Costea P.I., Cruaud C., d'Oviedo F., Engelen S., Ferrera I., Gasol J., Guidi L., Hildebrand F., Kokoszka F., Lepoivre C., Lima-Mendez G., Poulain J., Poulos B., Royo-Llonch M., Sarmento H., Vieira-Silva S., Dimier C., Picheral M., Searson S., Kandels-Lewis S., *Tara* Oceans coordinators, Bowler C., de Vargas C., Gorsky G., Grimsley N., Hingamp P., Iudicone D., Jaillon O., Not F., Ogata H., Pesant S., Speich S., Stemmann L., Sullivan M., Weissenbach J., Wincker P., Karsenti E., Raes J., Acinas S. and Bork P. (2015). Structure and function of the global ocean microbiome. *Science*, **348**, 6237.

Source

The raw data were downloaded from `http://ocean-microbiome.embl.de/companion.html`.

---

ukfs *UKFS*

Description

Select variables using unsupervised kernel method.

Usage

```
ukfs(X, kernel.func=c("linear", "gaussian.radial.basis", "bray"),
     method=c("kernel", "kpca", "graph"),
     keepX=NULL, lambda=NULL, n_components=2, Lg=NULL,
     mu=1, max_iter=100, ...)
```

Arguments

`X`	a numeric matrix (or data frame) used to select variables. `NA`s not allowed.
`kernel.func`	the kernel function name to use. Widely used kernel functions are pre-implemented, that can be used by setting `kernel.func` to one of the following strings: `"linear"`, `"gaussian.radial.basis"` or `"bray"`. Default: `"linear"`.
`method`	the method to use. Either an unsupervised variable selection method (`"kernel"`), a kernel PCA oriented variable selection method (`"kpca"`) or a structure driven variable selection selection (`"graph"`). Default: `"kernel"`.
`keepX`	the number of variables to select.
`lambda`	the penalization parameter that controls the trade-off between the minimization of the distorsion and the sparsity of the solution parameter.
`n_components`	how many principal components should be used with method `"kpca"`. Required with method `"kpca"`. Default: `2`.
`Lg`	the Laplacian matrix of the graph representing relations between the input dataset variables. Required with method `"graph"`.

`mu`	the penalization parameter that controls the trade-off between the the distorsion and the influence of the graph. Default: `1`.
`max_iter`	the maximum number of iterations. Default: `100`.
`...`	the kernel function arguments. In particular `sigma` (`"gaussian.radial.basis"`): double. The inverse kernel width used by `"gaussian.radial.basis"`.

Value

`ukfs` returns a vector of size.

Author(s)

Jerome Mariette <jerome.mariette@inra.fr>

Nathalie Villa-Vialaneix <nathalie.villa-vialaneix@inra.fr>

References

Mariette J., Brouard C. Flamary R. and Vialaneix N. (2020). Unsupervised variable selection for kernel methods in systems biology. *Preprint*

See Also

`compute.kernel`

Examples

```
data("Koren.16S")
## Not run:
ukfs.res <- ukfs(Koren.16S$data.raw, kernel.func = "bray", lambda=1)

## End(Not run)
```

Index

∗Topic **datasets**
    TARAoceans, 11

cim.kernel, 2
combine.kernels, 3, *6, 7*
compute.kernel, *3, 4*, 5, *7*, *9*, *13*
corrplot, *2*

kernel.pca, *4*, *6*, 7, *8–11*
kernel.pca.permute, 8, *10, 11*

mixKernel.users.guide, 9

plotVar.kernel.pca, *8, 9*, 10

TARAoceans, 11

ukfs, 12