Chemistry Aware Model Builder (camb). Package documentation

October 28, 2014

`camb` *Chemically Aware Model Builder (camb): An R package for property and bioactivity modeling of small molecules*

Description

camb allows molecule standardisation, descriptor generation, model building, model ensembling and new molecule predictions to be done within the same environment. Two comprehensive tutorials are also provided with the package. We strongly encourage camb users to read them.

References

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>. Chemically Aware Model Builder (camb): An R package for property and bioactivity modeling of small molecules.

`AADescs` *Amino Acid Descriptor Calculation*

Description

The function calculates amino acid descriptors for natural amino acids. Currently available descriptors are: 3 and 5 z-scales ("Z3" and "Z5"), T-scales ("TScales"), ST-scales ("STScales"), principal components score Vectors of Hydrophobic, Steric, and Electronic properties ("VHSE"), BLOSUM ("BLOSUM"), FASGAI ("FASGAI"), MSWHIM ("MSWHIM"), and ProtFP ("ProtFP8"). See references for further information on these descriptors.

Usage

```
AADescs(Data, type = "Z5", ...)
```

Arguments

Data	A character, vector, matrix or data.frame containing the amino acids in either one-letter or three-letter format. Amino acids symbols are valid in capitals or in lower-case. Gaps in alignments are expected to be represented with the charachter "-". The value of any of the descriptors provided for "-" is 0.
type	Type of descriptors to be calculated. Default value is 5 z-scales. Any combination of descriptors is valid. A vector containing the abbreviation of the desired descriptors is taken as argument.

Value

A data.frame which columns are indexed by the descriptors, and rows by the rows amino acid sequence. If the input data is a matrix or data.frame, the number of rows in the original data.frame or matrix, and the number of rows of the ouput data.frame are equal. If several descriptor types are chosen, descriptors are concatenated for the ease of further modeling. Column names indicate the amino acid position in the original input data, and the type of descriptor.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

References

http://www.jcheminf.com/content/5/1/41

http://www.jcheminf.com/content/5/1/42

Examples

```
AADescs(c("A","A"))
```

CorrelationPlot *Scatterplot of the Observed against the Predicted Values.*

Description

The function depicted a scatterplot of the observed against the predicted values with a machine learning model.

Usage

```
CorrelationPlot(pred, obs, margin = NULL, main = "", ylab = "Predicted",
xlab = "Observed", PointSize = 4, ColMargin = "blue", TextSize = 15,
TitleSize = 15, XAxisSize = 15, YAxisSize = 15, TitleAxesSize = 15,
tmar = 1, bmar = 1, rmar = 1, lmar = 1, AngleLab = 30,
LegendPosition = "right", PointColor = "black", PointAlpha = 1,
PointShape = 16, MarginWidth = 1)
```

Arguments

pred	Predicted values.
obs	Observed values.
margin	Bioactivity margin centered in the diagonal of the correlation plot. Default value NULL.
main	Plot title.
ylab	Title of the Y axis.
xlab	Title of the X axis.
PointSize	Size of the points.
ColMargin	Color of the bioactivity margin ('margin').
TextSize	Text font size. Default value 15.
TitleSize	Title font size. Default value 15.
XAxisSize	Size of the text on the X axis. Default value 15.
YAxisSize	Size of the text on the Y axis. Default value 15.
TitleAxesSize	Font size of the axes lables. Default value 15.
tmar	Top margin size. Default values is 1.
bmar	Bottom margin size. Default values is 1.
rmar	Right margin size. Default values is 1.
lmar	Left margin size. Default values is 1.
AngleLab	Angle of the labels in the X axis. Default value 30.
LegendPosition	Position of the legend. Default value 'right'.
PointColor	Color of the points in the scatterplot. Default value 'black'.
PointAlpha	Color alpha of the points in the scatterplot. Default value 1.
PointShape	Shape of the points in the scatterplot. Default value 16.
MarginWidth	

Value

A list (ggplot2 plot) with the scatterplot.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

Examples

```
CorrelationPlot(pred=seq(1,10)+rnorm(10), obs = seq(1,10),
margin = NULL, main = "", ylab = "Predicted",
    xlab = "Observed", PointSize = 4, ColMargin = "blue", TextSize = 15,
    TitleSize = 15, XAxisSize = 15, YAxisSize = 15, TitleAxesSize = 15,
    tmar = 1, bmar = 1, rmar = 1, lmar = 1, AngleLab = 30, LegendPosition = "right",
    PointColor = "black", PointAlpha = 1, PointShape = 16, MarginWidth = 1)
```

DensityResponse *Plot Distribution of the Response Variable*

Description

Plots the distribution of the response variable using a histogram.

Usage

```
DensityResponse(Data, xlab = "", ylab = "", main = "", alpha = 0.2,
  binwidth = NULL, histFill = "white", histCol = "black",
  densityFill = "#FF6666", TitleSize = 15, TextSize = 15,
  XAxisSize = 15, YAxisSize = 15, AngleLab = 30,
  LegendPosition = "right", TitleAxesSize = 15, tmar = 1, bmar = 1,
  rmar = 1, lmar = 1)
```

Arguments

Data	A numeric vector
xlab	Title of the x axis.
ylab	Title of the y axis.
main	Title of the plot.
alpha	Alpha for the fill color of the distribution. Default value 0.2.
binwidth	Width of the histogram bins. Default value NULL.
histFill	Fill color of the histogram bars. Default value 'white'.
histCol	Color of the histogram lines. Default value 'black'.
densityFill	Fill color of the distribution. Default value "#FF6666".
TitleSize	Title font size. Default value 15.
TextSize	Text font size. Default value 15.
XAxisSize	Size of the text on the X axis. Default value 15.
YAxisSize	Size of the text on the Y axis. Default value 15.
AngleLab	Angle of the labels in the X axis. Default value 30.
LegendPosition	Position of the legend. Default value 'right'.
TitleAxesSize	Font size of the axes lables. Default value 15.
tmar	Top margin size. Default values is 1.
bmar	ottom margin size. Default values is 1.
rmar	Right margin size. Default values is 1.
lmar	Left margin size. Default values is 1.

Details

Additional ggplot2 layers can be added with "+".

Value

Returns a ggplot object.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

DrawMoleculeInSDF *2D chemical structure visualization*

Description

DrawMoleculeInSDF permits the depiction of 2D chemical structures. See the tutorials for examples.

Usage

```
DrawMoleculeInSDF(structures.file, structure.number, file.name, useNameAsTitle)
```

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

ErrorBarplot *Barplot with error bars.*

Description

The function creates easily customizable barplots with error bars.

Usage

```
ErrorBarplot(X, Y, err, fill = X, main = "", ylab = "", xlab = "", TextSize = 15, TitleSize = 15
```

Arguments

X	A vector containing the aesthetics corresponding to the X axis.
Y	The values for the ordenate axis.
err	The standard deviation corresponding to the Y values.
fill	The groups that will be used to color the bars. The groups defined in the X variable are used by default.
main	Plot title.
ylab	Title of the Y axis.
xlab	Title of the X axis.
TextSize	Text font size. Default value 15.
TitleSize	Title font size. Default value 15.
XAxisSize	Size of the text on the X axis. Default value 15.
YAxisSize	Size of the text on the Y axis. Default value 15.
TitleAxesSize	Size of the title of both the X and Y axis. Default value 15.
AngleLab	Angle of the labels in the X axis. Default value 30.

barcol	Color of the error bars. Default value "red".
barSize	Size of the error bars. Default value 1.
barWidth	Width of the error bars. Default value 0.3.
LegendName	Name of the legend.
ColLegend	Number of columns of the legend. Default value 1.
RowLegend	Number of rows of the legend. Default value NULL - as many as groups-.
LegendPosition	Position of the legend. Default value "right".
tmar	Top margin size. Default values is 1.
bmar	Bottom margin size. Default values is 1.
rmar	Right margin size. Default values is 1.
lmar	Left margin size. Default values is 1.
stat	Default value "identity".

Value

A list containing the barplot (ggplot2 object).

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

Examples

```
d = data.frame(Y=seq(1,4),err=rep(1,4),X=c("A","B","C","D"))

# Example 1
ErrorBarplot(d$X,d$Y,d$err,fill=d$X,
main = "", ylab = "", xlab = "",
TextSize = 15, TitleSize = 15, XAxisSize = 15, YAxisSize = 15,
TitleAxesSize = 15, AngleLab = 35, barcol = "red", barSize = 1,
barWidth = 0.3, LegendName = "Legend", ColLegend = 1, RowLegend = NULL,
LegendPosition = "right", tmar = 1, bmar = 1, rmar = 1, lmar = 1,
stat = "identity")


# Example 2
ErrorBarplot(d$X,d$Y,d$err,fill=d$X,
main = "Example 2 ErrorBarplot", ylab = "Value", xlab = "Group",
TextSize = 15, TitleSize = 15, XAxisSize = 15, YAxisSize = 15,
TitleAxesSize = 15, AngleLab = 0, barcol = "green", barSize = 1,
barWidth = 0.6, LegendName = "Example Legend", ColLegend = 1,
RowLegend = NULL, LegendPosition = "right",
tmar = 1, bmar = 1, rmar = 1, lmar = 1,
stat = "identity")
```

GeneratePadelDescriptors *GeneratePadelDescriptors*

Description

Utilises the PaDEL-Descriptor Java library to generate molecular descriptors.

Usage

```
GeneratePadelDescriptors(standardised.file, types = c("2D"), threads = -1,
  limit = -1)
```

Arguments

standardised.file	The name of the file to which the standardised molecules were written to with the StandardiseMolecules function. If standardisation is not used then this can be any file in the SDF format.
types	The types of descriptors calculated. Options include: "2D", "Fingerprinter", "ExtendedFingerprinter", "EStateFingerprinter", "GraphOnlyFingerprinter", "MACCSFingerprinter,"PubchemFingerprinter","SubstructureFingerprinter", "SubstructureFingerprintCount","KlekotaRothFingerprinter" and "KlekotaRothFingerprintCount"

Value

A data.frame containing the descriptors with the prefix removed.

GetCVTrainControl *Sets up the control parameters of caret's* train *function.*

Description

Calls caret's trainControl function to set up the parameters of the train function. This control variable is saved into the dataset list as dataset$trControl.

Usage

```
GetCVTrainControl(dataset, seed = 1, folds = 5, repeats = 1,
  method = "cv", returnResamp = "none", returnData = TRUE,
  savePredictions = TRUE, verboseIter = TRUE, allowParallel = TRUE, ...)
```

Arguments

dataset	The training dataset returned by SplitSet
seed	The seed for randomization so that the fold selection can be done in a repeatable way if desired
folds	The number of folds to use during cross-validation
repeats	For repeated k-fold cross-validation only: the number of complete sets of folds to compute
returnResamp	A character string indicating how much of the resampled summary metrics should be saved. Values can be final, all or none
returnData	A logical for saving the data
savePredictions	A logical to save the hold-out predictions for each resample
verboseIter	A logical for printing a training log.
allowParallel	If a parallel backend is loaded and available, should the function use it?

Value

a dataset with the traincontrol saved within for future training

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

GetPropertiesSDF *Extract the properties values for the molecules in an input '.sdf' file.*

Description

This function returns a data.frame containing the values for all properties for a user-defined number of molecules from an input '.sdf' file.

Usage

```
GetPropertiesSDF(structures.file, number_processed = -1)
```

Arguments

structures.file	Input '.sdf' file.
number_processed	The number of molecules which properties should be returned. The properties values for all molecules are returned when using the default value, namely -1.

Value

A data.frame containing the values for the properties.

Author(s)

Isidro Cortes-Ciriano <isidrolauscher@gmail.com>

See Also

ShowPropertiesSDF GetPropertySDF

`GetPropertySDF` *Extract the values corresponding to a given property for the molecules from an input '.sdf' file.*

Description

This function returns a data.frame containing the values for a selected property for a user-defined number of molecules from an input '.sdf' file.

Usage

```
GetPropertySDF(structures.file, property = "", number_processed = -1)
```

Arguments

`structures.file`
: Input '.sdf' file.

`property`
: Property name (in string format) which values should be returned.

`number_processed`
: The number of molecules for which the propertt values should be returned. The property values for all molecules are returned when using the default value, namely -1.

Value

A data.frame containing the value for the selected property (argument `property`)

Author(s)

Isidro Cortes-Ciriano <isidrolauscher@gmail.com>

See Also

GetPropertiesSDF ShowPropertiesSDF

ImputeFeatures *Impute missing descriptor values using knn.impute*

Description

A nearest neighbour based

Usage

```
ImputeFeatures(d, k = 10, ...)
```

Arguments

d	A data.frame

Value

A data.frame with infinite values replaced by NA.

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

MAE *Mean Absolute Error (MAE)*

Description

Mean Absolute Error (MAE) between two numerical input vectors.

Usage

```
MAE(v1, v2)
```

Arguments

v1	Input vector. e.g. the predicted values for the dependent variable.
v2	Input vector. e.g. the observed values for the dependent variable.

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

MaxPerf *Distribution of Maximum Theoretical Values Achievable given the Dataset and its Uncertainty*

Description

Calculates the ditribution of model validation metrics that are achievable given the size of the dataset, the uncertainty in the response variable, and the distribution of the responsible variable quantified by its mean and standard deviation. Therefore, these distributions help to assess models overfitting; e.g. a model trained on a dataset with high uncertainty exhibiting high correlation values might be overoptimistic. See the tutorials for more information and examples.

Usage

```
MaxPerf(meanNoise = 0, sdNoise, resp, lenPred, stds = NULL,
    iters = 1000, filename = NULL, pdfW = 10, pdfH = 10, TextSize = 15,
    TitleSize = 15, XAxisSize = 15, YAxisSize = 15, TitleAxesSize = 15,
    tmar = 1, bmar = 1, rmar = 1, lmar = 1, AngleLab = 30,
    LegendPosition = "right")
```

Arguments

meanNoise	Mean value of the noise in the data. Default value 0.
sdNoise	Standard deviation of the noise in the data. See the work by Kramer et al. about uncertainty in public bioactivity databases.
resp	Vector containing the values for the dependent variable in the dataset (y).
lenPred	Number of datapoints of the external (hold-out) set.
stds	Vector containing the experimental error for the response variable. The default value is NULL.
iters	Number of iterations. Default value 1000.
filename	If not NULL, file where the plot will be saved. Default value NULL.
pdfW	Width of the .pdf file, in centimeters, where the plot will be saved. Default value 10.
pdfH	Height of the .pdf file, in centimeters, where the plot will be saved. Default value 10.
TextSize	Fontsize of the text in the plot. Default value 15.
TitleSize	Fontsize of the title. Default value 15.
XAxisSize	Fontsize of the X axis. Default value 15.
YAxisSize	Fontsize of the Y axis. Default value 15.
TitleAxesSize	Fontsize of the axes titles. Default value 15.
tmar	Top margin size. Default value 1.
bmar	Bottom margin size. Default value 1.
rmar	Right margin size. Default value 1.
lmar	Left margin size. Default value 1.
AngleLab	Angle of the labels of the X axis. Default value 30.
LegendPosition	Position of the legend. Default value 'right'.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

References

Cortes-Ciriano et al. 'Proteochemometric Modeling in a Bayesian Framework'. J. Cheminf. 6, 35. 2014 http://www.jcheminf.com/content/6/1/35

`MinPerf` *Distribution of Minimum Theoretical Values Achievable given the Dataset and its Uncertainty*

Description

Similar to MaxPerf, with the exception that the dependent variable is randomized before calculating the statistical metrics. For futher information see the tutorials and Cortes-Ciriano et al. 'Proteochemometric Modeling in a Bayesian Framework'. J. Cheminf. 6, 35. 2014 http://www.jcheminf.com/content/6/1/35

Usage

```
MinPerf(meanNoise = 0, sdNoise, resp, lenPred, stds = NULL, iters = 1000,
  filename = NULL, pdfW = 10, pdfH = 10, TextSize = 15,
  TitleSize = 15, XAxisSize = 15, YAxisSize = 15, TitleAxesSize = 15,
  tmar = 1, bmar = 1, rmar = 1, lmar = 1, AngleLab = 30,
  LegendPosition = "right")
```

Arguments

`meanNoise`	Mean value of the noise in the data. Default value 0.
`sdNoise`	Standard deviation of the noise in the data. See the work by Kramer et al. about uncertainty in public bioactivity databases.
`resp`	Vector containing the values for the dependent variable in the dataset (y).
`lenPred`	Number of datapoints of the external (hold-out) set.
`stds`	Vector containing the experimental error for the response variable. The default value is NULL.
`iters`	Number of iterations. Default value 1000.
`filename`	If not NULL, file where the plot will be saved. Default value NULL.
`pdfW`	Width of the .pdf file, in centimeters, where the plot will be saved. Default value 10.
`pdfH`	Height of the .pdf file, in centimeters, where the plot will be saved. Default value 10.
`TextSize`	Fontsize of the text in the plot. Default value 15.
`TitleSize`	Fontsize of the title. Default value 15.
`XAxisSize`	Fontsize of the X axis. Default value 15.
`YAxisSize`	Fontsize of the Y axis. Default value 15.
`TitleAxesSize`	Fontsize of the axes titles. Default value 15.

tmar	Top margin size. Default value 1.
bmar	Bottom margin size. Default value 1.
rmar	Right margin size. Default value 1.
lmar	Left margin size. Default value 1.
AngleLab	Angle of the labels of the X axis. Default value 30.
LegendPosition	Position of the legend. Default value 'right'.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

MorganFPs *Circular Morgan Fingerprints as specified in RDkit*

Description

The function calculates circular Morgan fingerprints for chemical compounds using the RDkit python library (Greg Landrum). Hashed fingeprints are calculated in binary format or with counts. In addition, it also calcualates unhashed fingerprints, both binary and with counts.

Usage

```
MorganFPs(bits = 512, radius = 2, type = "smi", mols, output,
  keep = "hashed_binary", images = FALSE, unhashed = FALSE,
  verbose = FALSE, RDkitPath = "/usr/local/share/RDKit",
  PythonPath = "/usr/local/lib/python2.7/site-packages",
  extFileExtension = FALSE, extMols = FALSE, unhashedExt = FALSE,
  logFile = FALSE)
```

Arguments

bits	Number of bits of the hashed fingerprints.
radius	Radius of the hashed fingerprints. A radius of 2 is equivalent to ECFP-4 fingerprints.
type	File format containing the input molecules.
mols	File containing the input molecules.
output	Labels that will be appended to all ouput files.
keep	The fingeprints that will be kept after the calculation. Apart from calculating different types of fingerprints, the function returns a data.frame with the type of fingerprints specified here. Possible types are: hashed_binary, hashed_counts, unhashed_binary, unhashed_counts, hashed_binaryEXT, hashed_countsEXT, unhashed_binaryEXT and unhashed_countsEXT.
images	If TRUE individual .pdf files containing the image of each substructure present in the input file, and for each molecule, are created. Be aware that the number of fingerprints can be large depending on the number and diversity of the molecules present in the input file. Thus, allow for sufficient memory.
unhashed	If TRUE, unhashed fingeprints -both in binary format and with counts- are calculated.

verbose	If TRUE, information about the progression of the calculation is printed.
RDkitPath	The path to the folder containing the RDkit library in your computer. On mac, the is equal to the environment variable $RDBASE.
PythonPath	Path to python ($PYTHONPATH).
extFileExtension	File extension for the file containing the molecules for which unhashed fingeprints are to be calculated with respect to the pool of substructures in the molecules present in the file specified in 'mols'.
extMols	File containing the molecules for which unhashed fingeprints are to be calculated with respect to the pool of substructures in the molecules present in the file specified in 'mols'.
unhashedExt	If TRUE, unhashed fingerprints are calcualted for the molecules in 'extMols'.
logFile	File where the log messages will be dropped.

Value

In the working directory, .csv files will be created containing the different fingeprint types specified with the function arguments. By default, hashed fingerprint in binary format and with counts will be created. In addition, the function returns a data.frame with the fingerprint types defined in the argument 'keep'. In the data.frame, rows are indexed by the molecules in the file containing the molecules, and columns by the bits in the fingerprint vector.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

References

FingeprintCalculator.py. Isidro Cortes. 2013/2014. http://github.com/isidroc/FingerprintCalculator.

Examples

```
test_mols <- system.file("test_structures", "structures_10.sdf", package = "camb")
MorganFPs(bits=28,radius=4,type="sdf",mols=test_mols,output="test_mols")

#See the camb tutorials for more examples.
```

PCA *Principal Component Analysis (PCA)*

Description

The function "PCA" enables the calculation of the Principal Components (PCs) for a given set of descriptors. The function takes as arguments the descriptors and, optionally, the names of the rows, i.e. datapoints. Further arguments of the function prcomp from the package stats, use to run the PCA analysis, can be additionally set. The function returns a list with following elements :

Data : a dataframe containing the two first PCs and the row names if provided. Rows are indexed as in the input data corresponds to a list. PCs_All| : a dataframe containing all PCs. Std : a vector containing the standard deviation of all PCs. Info : information about the PCA analysis, such as the

proportion of variance explained by each PC. It is always advisable to verify that the two or three first PCs explain a large proportion of the variance in the data, if conclusions are to be extracted from this type of analysis.

Usage

```
PCA(Data, RowNames = NULL, cor = TRUE, scale = TRUE, center = TRUE, ...)
```

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

PCAPlot *Plot PCA analysis*

Description

The function "PCAPlot" provides an easy way to plot the first two PC. As all the plotting function provided with camb, it is based on ggplot2, which allows further customization by the user. Below is an example of how to use these two function to do a PCA analysis of the target space, which in this case is quantified by the amino acid descriptors of the amino acids present in the binding site of mammal cyclooxygenases.

Usage

```
PCAPlot(Data, labels = NULL, Colors = NULL, Shapes = NULL,
  main = "", ylab = "PC2", xlab = "PC1",
  PointSize = 4, LegendPosition = "right", LegendName = "",
  ColLegend = 1, RowLegend = NULL, TitleSize = 15, TextSize = 15,
  XAxisSize = 15, YAxisSize = 15, AngleLab = 0, TitleAxesSize = 15,
  LegendTitleSize = 15, LegendTextSize = 15, tmar = 1, bmar = 1,
  rmar = 1, lmar = 1)
```

PairwiseDist *Pairwise Distance (Similarity) Matrix*

Description

The function is based on the vegdist function from the vegan package. It calculated the pairwise distance similarity matrix for all vectors input in a matrix or data.frame. The functions operates on a row basis.

Usage

```
PairwiseDist(Data, method = "jaccard", ...)
```

Arguments

Data	A numeric data.frame or matrix containing compound, protein or amino acid descriptors (or any combination thereof).
method	Available distance metrics are: "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao", "cao". See the documentation of the R package vegan for details.

Details

For further detials see the documentation in the R package vegan.

Value

A data.frame with the all pairwise distances.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

See Also

PairwiseDistPlot

Examples

```
m = matrix(abs(rnorm(20)),4,4)
mDist = PairwiseDist(m)
head(mDist)
```

PairwiseDistPlot *Distribution of Pairwise Similarities*

Description

The function depicts the distribution of pairwise similarities.

Usage

```
PairwiseDistPlot(Data, xlab = "", ylab = "", main = "", TextSize = 15,
TitleSize = 15, XAxisSize = 15, YAxisSize = 15, TitleAxesSize = 15, tmar = 1,
bmar = 1, rmar = 1, lmar = 1, AngleLab = 30, binwidth = NULL, fillCol = "white",
 Colour = "black", DensityFill = "#FF6666", DensityAlpha = 0.2)
```

Arguments

Data	A data.frame with a single column named 'Distance'. This is the default output of PairwiseDist.
xlab	Label of the X axis.
ylab	Label of the Y axis.
main	Plot title.
TextSize	Fontsize of the text in the plot. Default value 15.
TitleSize	Fontsize of the title. Default value 15.
XAxisSize	Fontsize of the X axis. Default value 15.
YAxisSize	Fontsize of the Y axis. Default value 15.
TitleAxesSize	Fontsize of both the X and Y axes. Default value 15.
tmar	Top margin size. Default value 1.
bmar	Bottom margin size. Default value 1.
rmar	Right margin size. Default value 1.
lmar	Left margin size. Default value 1.
AngleLab	Angle of the labels in the X axis. Default value 30.
binwidth	Width of the bins of the hitogram. Default NULL, which corresponds to 1/30 of the range of the data (see ??stat_bin of ggplot2).
fillCol	Fill color of the histogram. Default value 'white'.
Colour	Line color of the histogram. Default value 'black'.
DensityFill	Fill color of the distribution. Default value "#FF6666".
DensityAlpha	Alpha for the fill color of the distribution. Default value 0.2.

Value

A ggplot2 object with the pairwise distance distribution.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

References

Package ggplot2.

See Also

PairwiseDist

Examples

```
m <- matrix(abs(rnorm(1600)),40,40)
mDist <- PairwiseDist(m)
head(mDist)
mDistPlot <- PairwiseDistPlot(mDist,xlab = "", ylab = "", main = "", TextSize = 15,
TitleSize = 15, XAxisSize = 15, YAxisSize = 15, TitleAxesSize = 15, tmar = 1, bmar = 1,
 rmar = 1, lmar = 1, AngleLab = 30, binwidth = NULL, fillCol = "white",
  Colour = "black", DensityFill = "#FF6666", DensityAlpha = 0.2)
```

`PlotMolecules` *Plot Compounds from a .sdf File.*

Description

The function plots the chemical structures provided in a .sdf file. The plots can also be sent to a 2x2 grid in a .pdf file.

Usage

```
PlotMolecules(sdf.file, IDs, pdf.file = NULL, PDFMain = NULL, useNameAsTitle = TRUE)
```

Arguments

`sdf.file`	The .sdf file with the molecules.
`IDs`	The IDs of the molecules to be depicted (the ordinal position of the molecules in the .sdf file). Currently, a maximum of four IDs is supported.
`pdf.file`	If not NULL, the .pdf where the molecules will be depicted.
`PDFMain`	If not NULL, the title of the molecule depiction in the .pdf file.
`useNameAsTitle`	If TRUE, the names of the molecules as especified in the .sdf file are used as molecules names in the depiction.

Value

A list with the plots of the molecules.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

`PreProcess` *Cals the* `preProcess` *function of the caret package which handles data transformation before training.*

Description

Pre-processing transformation (centering, scaling etc.) can be estimated from the training data and applied to any data set with the same variables. The simplest form of this is to center and scale all the variables so that each of their means are 0 and each of their standard devitations is 1. See `preProcess` arguments for more control.

Usage

```
PreProcess(dataset, steps = c(center, scale), ...)
```

Arguments

dataset	The training dataset returned by SplitSet
steps	A character vector specifying the type of processing. Possible values are 'BoxCox', 'YeoJohnson', 'expoTrans', 'center', 'scale', 'range', 'knnImpute', 'bagImpute', 'medianImpute', pca', 'ica' and 'spatialSign' for details see the preProcess function for more details.

Value

A dataset with the preprocessed training and holdout set that also stores the transformation

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

PredictExternal *Make predictions on new molcules using a saved standardisation procedure and a saved model*

Description

Molecules are converted to a standard representation in the same way as during model training. A saved model is used to make predictions on new molecules.

Usage

```
PredictExternal(structures.file, standardisation.options, descriptor.types,
  dataset, model)
```

Arguments

structures.file	The name of the file containing the chemical structures. SMILES and SDF are currently supported formats.
standardisation.options	The options saved during the standardisation procedure. These options are returned from the StandardiseMolecules function.
descriptor.types	A named list of the types of descriptors used in model training.
dataset	The dataset used in model training. This is used for the preprocessing applied to the training data as well as the descriptors used in training.
model	The trained model.

Value

A data.frame containing the original ids of the molecules as well as their predicted values.

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

Examples

```
#test_structures_file <- system.file("test_structures", "structures_10.sdf", package = "camb")
# The following requires a trained model, e.g. rf.rds
#predictions <- PredictExternal(test_structures_file,
#standardisation.options, descriptor.types, dataset, readRDS("rf.rds"))
```

`Qsquared1` *Q squared 1*

Description

Calculates the Q squared 1 for two input vectors, e.g. the observed and the predicted values for a test set. See the function Validation and the camb tutorials for further information.

Usage

```
Qsquared1(v1, v2, resp_tr)
```

Arguments

`v1`	Input vector. e.g. the predicted values for the dependent variable.
`v2`	Input vector. e.g. the observed values for the dependent variable.
`resp_tr`	Vector containing the values of the dependent variable corresponding to the datapoints in the training set.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

`Qsquared2` *Calculates the Q Squared 2*

Description

Calculates the Q squared 2 for two input vectors, e.g. the observed and the predicted values for a test set. See the function Validation and the camb tutorials for further information.

Usage

```
Qsquared2(v1, v2)
```

Arguments

`v1`	Input vector. e.g. the predicted values for the dependent variable.
`v2`	Input vector. e.g. the observed values for the dependent variable.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

Qsquared3 *Calculates the Q Squared 3*

Description

Calculates the Q squared 3 for two input vectors, e.g. the observed and the predicted values for a test set. See the function Validation and the camb tutorials for further information.

Usage

```
Qsquared3(v1, v2, resp_tr)
```

Arguments

v1	Input vector. e.g. the predicted values for the dependent variable.
v2	Input vector. e.g. the observed values for the dependent variable.
resp_tr	Vector containing the values of the dependent variable corresponding to the datapoints in the training set.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

RMSE *RMSE*

Description

Calculates the RMSE for two input vectors, e.g. the observed and the predicted values for a test set. See the function Validation and the camb tutorials for further information.

Usage

```
RMSE(v1, v2)
```

Arguments

v1	Input vector. e.g. the predicted values for the dependent variable.
v2	Input vector. e.g. the observed values for the dependent variable.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

RMSE_CV *Extract the cross validated RMSE (or another metric) from a caret model*

Description

Extract the value of a cross-validated metric (e.g. RMSE) from a caret model

Usage

```
RMSE_CV(model, digits = 3, metric = "RMSE")
```

Arguments

mode	Trained caret model from which the value for the cross-validated metric specified in metric.
digits	Number of decimals to which the value is to be rounded.
metric	Cross-validated metric which value is to be obtained. Default value is 'RMSE'.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

RemoveHighlyCorrelatedFeatures *Calls the* findCorrelation *function of the caret package which finds the highly correlated descriptors and removes them from the training and holdout sets.*

Description

The absolute values of pair-wise correlations are considered. If two variables have a high correlation, the function looks at the mean absolute correlation of each variable and removes the variable with the largest mean absolute correlation.

Usage

```
RemoveHighlyCorrelatedFeatures(dataset, correlationCutoff = 0.95, ...)
```

Arguments

dataset	The training dataset returned by SplitSet
correlationCutoff	A numeric value for the pair-wise absolute correlation cutoff

Value

A dataset with the appropriate columns cut out of the x.train and x.holdout dataframes

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

RemoveNearZeroVarianceFeatures

Calls the nearZeroVar *function of the caret package and removes descriptos with near zero variance in the appropriate columns from the training and holdout sets.*

Description

nearZeroVar diagnoses predictors that have one unique value (i.e. are zero variance predictors) or predictors that are have both of the following characteristics: they have very few unique values relative to the number of samples and the ratio of the frequency of the most common value to the frequency of the second most common value is large. checkConditionalX looks at the distribution of the columns of x conditioned on the levels of y and identifies columns of x that are sparse within groups of y.

Usage

```
RemoveNearZeroVarianceFeatures(dataset, frequencyCutoff = 30/1, ...)
```

Arguments

dataset	The training dataset returned by SplitSet
frequencyCutoff	The cutoff for the ratio of the most common value to the second most common value

Value

A dataset with the appropriate columns cut out of the x.train and x.holdout dataframes

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

RemoveStandardisedPrefix

Remove the prefix that was added to the molecule names before standardisation.

Description

This function is needed to remove the prefix that is added to the molecule names to make sure that they don't start with a number. PaDEL-Descriptor doesn't handle molecule names starting with certain characters so a prefix is added before its use and then removed with this function. This function will be removed at some point when the PaDEL-Descriptor issue is resolved.

Usage

```
RemoveStandardisedPrefix(descriptors)
```

Arguments

descriptors	A data.frame containing the descriptors.

Value

A data.frame containing the descriptors with the prefix removed.

ReplaceInfinitesWithNA *Remove infinite values from the descriptor data.frame*

Description

Any infinites found in the descriptor data.frame are replaced with NA.

Usage

```
ReplaceInfinitesWithNA(d)
```

Arguments

d	A data.frame

Value

A data.frame with infinite values replaced by NA.

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

Rsquared *Rsquared*

Description

Calculates the R squared for two input vectors, e.g. the observed and the predicted values for a test set. See the function Validation and the camb tutorials for further information.

Usage

```
Rsquared(v1, v2)
```

Arguments

v1	Input vector. e.g. the predicted values for the dependent variable.
v2	Input vector. e.g. the observed values for the dependent variable.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

`Rsquared0` *Rsquared0*

Description

Calculates the R squared 0 for two input vectors, e.g. the observed and the predicted values for a test set. See the function Validation and the camb tutorials for further information.

Usage

```
Rsquared0(v1, v2)
```

Arguments

`v1`	Input vector. e.g. the predicted values for the dependent variable.
`v2`	Input vector. e.g. the observed values for the dependent variable.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

`Rsquared_CV` *Calculates the cross validated RSquared from a caret model*

Description

Calculate the cross validated RSquared

Usage

```
Rsquared_CV(model, digits = 3)
```

Arguments

`model`	Trained caret model from which the cross-validated R squared is to be obtained.
`digits`	Number of decimals to which the value is to be rounded.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

SeqDescs *Whole Protein Sequence Descriptor Calculation*

Description

Calculation of the following 12 whole sequence protein descriptors: Amino Acid Composition ("AAC"), Dipeptide Composition ("DC"), Tripeptide Composition ("TC"), Normalized Moreau-Broto Autocorrelation ("MoreauBroto"), Moran Autocorrelation ("Moran"), Geary Autocorrelation ("Geary") , CTD (Composition/Transition/Distribution) ("CTD"), Conjoint Traid ("CTriad"), Sequence Order Coupling Number ("SOCN"), Quasi-sequence Order Descriptors ("QSO"), Pseudo Amino Acid Composition ("PACC"), Amphiphilic Pseudo Amino Acid Composition ("APAAC").

Usage

```
SeqDescs(data, UniProtID = TRUE, type = "AAC", ...)
```

Arguments

data	One or more protein sequences, or one or several UniProt IDs.
UniProtID	If TRUE the argument calculates the descriptors for the proteins which UniProt IDs have been indicated in the argument 'data'.
type	The type of protein descriptors to be calculated (see above). Any combination thereof is valid. A vector containing the abbreviation of the desired descriptors is taken as argument. Default value 'AAC'.

Value

A numeric matrix which rows are indexed by proteins and the columns by descriptors. If multiple descriptors are chosen, the function returns a matrix where descriptors are concatenated per row for the ease of modeling.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

References

R package protr

ShowPropertiesSDF *Show the fields of a .sdf file*

Description

ShowPropertiesSDF returns a vector containing the names of the fields in an input '.sdf' file.

Usage

```
ShowPropertiesSDF(structures.file)
```

Arguments

structures.file	A '.sdf' file.

Value

A vector containing the field (properties) names present in the input '.sdf' file.

Author(s)

Isidro Cortes-Ciriano <isidrolauscher@gmail.com>

See Also

GetPropertiesSDF GetPropertySDF

Examples

```
# ShowPropertiesSDF("./test.sdf")
```

SplitSet *Split into training and hold out data*

Description

Creates a training/holdout split for the data

Usage

```
SplitSet(ids, x, y, percentage = 20, seed = 1)
```

Arguments

ids	The names of the molecules
x	The descriptors
y	The target values
percentage	The percentage of data to put into the holdout set
seed	The seed for randomization so that the split can be does in a repeatable way if required

Value

a list designed to be passed between functions in the camb workflow containing the variables required for training: ids, holdout.indexes, train.indexes, x.train, x.holdout, y.train, y.holdout

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

`StandardiseMolecules` *Convert molecules to a standard representation*

Description

Molecules are converted to a standard representation using Indigo's C API. Molecules can be read in either SMILES or SDF format. Hydrogens are made implicit. Molecules are excluded if they don't pass Indigo's checks for correctness which include incorrect valence representations and ambiguous Hydrogen representations. Atomic isotopes are converted to their common forms. Molecules are dearomatized and then converted to InChI format using Indigo's InChI plugin. Molecules are then converted back to a SMILES representation. Passing them through the InChI format essentially convert all tautomeric forms of the same molecule to a single representation. Various parameters are available to control which molecule get kept in the standardised set.

Usage

```
StandardiseMolecules(structures.file, standardised.file, removed.file = "",
  properties.file = "standardisation_info.csv", remove.inorganic = FALSE,
  fluorine.limit = -1, chlorine.limit = -1, bromine.limit = -1,
  iodine.limit = -1, min.mass.limit = -1, max.mass.limit = -1,
  number.processed = -1)
```

Arguments

`structures.file`
: The name of the file containing the chemical structures. SMILES and SDF are currently supported formats.

`standardised.file`
: The name of the file to which the standardised molecules are written to. This file is saved in the SDF format.

`removed.file`
: The name of the file to which the standardised molecules that were removed by the filters are written to. This file is saved in SDF format. If left out, this file is not created.

`properties.file`
: The name of the file to which the molecular properties contained in the `structures.file` are written to. This file is saved in CSV format.

`remove.inorganic`
: If set `TRUE`, molecules that contain any atoms not in H, C, N, O, P, S, F, Cl, Br, I are excluded.

`fluorine.limit`
: If specified, molecules with more than `flourine.limit` Flourine atoms are excluded.

`chlorine.limit`
: If specified, molecules with more than `chlorine.limit` Chlorine atoms are excluded.

`bromine.limit`
: If specified, molecules with more than `bromine.limit` Bromine atoms are excluded.

`iodine.limit`
: If specified, molecules with more than `iodine.limit` Iodine atoms are excluded.

`min.mass.limit`
: If specified, molecules with a molecular mass smaller than `min.mass.limit` are excluded.

`max.mass.limit` If specified, molecules with a molecule mass greater than `max.mass.limit` are excluded.

`number.processed`
If specified, only the first `number.processed` molecules will be processed by this function. This is used mainly for testing purposed on files that contain a lot of molecules.

Value

The options used in standardisation so that they may be applied to make predictions on new molecules using the function `PredictExternal`.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

References

http://www.ggasoftware.com/opensource/indigo

http://www.iupac.org/home/publications/e-resources/inchi.html

Examples

```
test_mols <- system.file("test_structures",
 "structures_10.sdf", package = "camb")
StandardiseMolecules(structures.file=test_mols,
 standardised.file="st.sdf", removed.file="removed.sdf",
 properties.file="properties.csv")
```

Validation *Statistical Metrics to Assess Model Peformance*

Description

The function calculates correlation and error metrics between two numeric vectors. These metrics are used to evaluate model performance on a test or external set.

Usage

```
Validation(pred, obs,resp_tr)
```

Arguments

`pred` A numeric vector with the predicted values.

`obs` A numeric vector with the observed values.

`resp_tr` The argument resp_tr, requires the bioactivity values of the datapoints present in the training set. These values are required by the metrics Q_1 and Q_3 (see camb tutorials).

Details

The predictive ability of the models on a test or validation set is evaluated by the calculation of the following statistical metrics: Q_1^2, Q_2^2, Q_3^2, RMSE, R^2, and R_0^2.

Value

A list containing the values for the statistical metrics. See the tutorial for the formula of the metrics, and for further details.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

References

camb tutorials.

Examples

```
# See the tutorials for examples.
```

`YScrambling` *Y-Scrambling*

Description

This function serves to randomize (permutate) a subset of the dependent variable. It is useful to perform Y-scrambling experiments.

Usage

```
YScrambling(y, percent)
```

Arguments

`y`	Input numerical vector.
`percent`	Percentage of the input vector to be randomized.

Value

A numerical vector with a user-defined percentage of values randomized.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

caretEnsemble *Combine several predictive models via weights*

Description

Find a good linear combination of several classification or regression models, using either linear regression, elastic net regression, or greedy optimization.

Usage

```
caretEnsemble(all.models, optFUN = NULL, ...)
```

Arguments

all.models	a list of caret models to ensemble.
optFUN	the optimization function to use
...	additional arguments to pass to the optimization function

Details

Every model in the "library" must be a separate `train` object. For example, if you wish to combine a random forests with several different values of mtry, you must build a model for each value of mtry. If you use several values of mtry in one train model, (e.g. tuneGrid = expand.grid(.mtry=2:5)), caret will select the best value of mtry before we get a chance to include it in the ensemble. By default, RMSE is used to ensemble regression models, and AUC is used to ensemble Classification models. This function does not currently support multi-class problems

Value

S3 caretEnsemble object

References

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.2859&rep=rep1&type=pdf

caretStack *Combine several predictive models via stacking*

Description

Find a good linear combination of several classification or regression models, using either linear regression, elastic net regression, or greedy optimization.

Usage

```
caretStack(all.models, ...)
```

Arguments

`all.models`	a list of caret models to ensemble.
`optFUN`	the optimization function to use
`...`	additional arguments to pass to the optimization function

Details

Every model in the "library" must be a separate `train` object. For example, if you wish to combine a random forests with several different values of mtry, you must build a model for each value of mtry. If you use several values of mtry in one train model, (e.g. tuneGrid = expand.grid(.mtry=2:5)), caret will select the best value of mtry before we get a chance to include it in the ensemble.

Value

S3 caretStack object

References

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.2859&rep=rep1&type=pdf

---

checkModels_extractTypes *Check that a list of models are all train objects and are ready to be ensembled together*

Description

Check that a list of models are all train objects and are ready to be ensembled together

Usage

```
checkModels_extractTypes(list_of_models)
```

Author(s)

Daniel Murrell <dsmurrell@gmail.com> and Isidro Cortes <isidrolauscher@gmail.com>

---

checkPreds *Check that a list of predictions from caret models are all valid*

Description

Check that a list of predictions from caret models are all valid

Usage

```
checkPreds(list_of_models)
```

Arguments

`list_of_models`	a list of caret models to check

expGrid *Exponential Grid Definition*

Description

The function defines an exponential series, which can be used, e.g. when defining the parameter space when training some models such as Support Vector Machines or Gaussian Processes.

Usage

```
expGrid(power.from, power.to, power.by, base)
```

Arguments

power.from	The starting exponential of the series.
power.to	The latest exponential of the series.
power.by	The exponential step of the series.
base	The base of the exponential series.

Value

A vector with the exponential series.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

Examples

```
expGrid(power.from=-10,power.to=10,power.by=2,base=10)
```

extractBestPreds *Extract predictions for the best tune from a list of caret models*

Description

Extract predictions for the best tune from a list of caret models

Usage

```
extractBestPreds(list_of_models)
```

Arguments

list_of_models	a list of caret models to extract predictions from

greedOptAUC *TODO*

Description

TODO

Usage

```
greedOptAUC(X, Y, iter = 100L)
```

Arguments

X
Y
iter

greedOptRMSE *TODO*

Description

TODO

Usage

```
greedOptRMSE(X, Y, iter = 100L)
```

Arguments

X
Y
iter

makePredObsMatrix *Extract obs from one models, and a matrix of predictions from all other models*

Description

Extract obs from one models, and a matrix of predictions from all other models

Usage

```
makePredObsMatrix(list_of_models)
```

Arguments

list_of_models	a list of caret models to extract predictions from

mergeData *Merge descriptors blocks.*

Description

The function merges blocks of descriptors by columns.

Usage

```
mergeData(a1, a2, a3, a4 = NULL, a5 = NULL, a6 = NULL)
```

Arguments

a1..a6	Descriptors blocks with the same number of rows to be merged.

Value

The merged block of descriptors.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

multiPredict *Make a matrix of predictions from a list of caret models*

Description

Make a matrix of predictions from a list of caret models

Usage

```
multiPredict(list_of_models, type, newdata = NULL, ...)
```

Arguments

list_of_models	a list of caret models to make predictions for
type	Classification or Regression
...	additional arguments to pass to predict.train. DO NOT PASS the "type" argument. Classsification models will returns probabilities if possible, and regression models will return "raw".

`predict.caretEnsemble` *Make predictions from a caretEnsemble. This function passes the data to each function in turn to make a matrix of predictions, and then multiplies that matrix by the vector of weights to get a single, combined vector of predictions.*

Description

Make predictions from a caretEnsemble. This function passes the data to each function in turn to make a matrix of predictions, and then multiplies that matrix by the vector of weights to get a single, combined vector of predictions.

Usage

```
predict.caretEnsemble(ensemble, ...)
```

Arguments

`ensemble`	a caretEnsemble to make predictions from.
`...`	arguments (including newdata) to pass to predict.train.

`predict.caretStack` *Make predictions from a caretStack. This function passes the data to each function in turn to make a matrix of predictions, and then multiplies that matrix by the vector of weights to get a single, combined vector of predictions.*

Description

Make predictions from a caretStack. This function passes the data to each function in turn to make a matrix of predictions, and then multiplies that matrix by the vector of weights to get a single, combined vector of predictions.

Usage

```
predict.caretStack(ensemble, newdata = NULL, ...)
```

Arguments

`ensemble`	a caretStack to make predictions from.
`...`	arguments (including newdata) to pass to predict.train.

slope *Slope*

Description

Slope between two vectors calculated as: sum(v2 * v1)/sum(v1 * v1) See the camb tutorials for further information.

Usage

```
slope(v1, v2)
```

Arguments

v1	Input vector. e.g. the predicted values for the dependent variable.
v2	Input vector. e.g. the observed values for the dependent variable.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com> and Daniel Murrell <dsmurrell@gmail.com>

Index