Package ‘svs’

January 13, 2016

Title Tools for Semantic Vector Spaces
Version 1.1.0
Description Various tools for semantic vector spaces, such as correspondence analysis (simple, multiple and discriminant), latent semantic analysis, probabilistic latent semantic analysis, non-negative matrix factorization, latent class analysis and EM clustering. Furthermore, there are specialized distance measures, plotting functions and some helper functions.
Depends R (>= 3.1.0), graphics, stats, utils
Imports gtools
Suggests MASS, pvclust
License GPL-3
LazyData true
Date 2016-01-13
Author Koen Plevoets [aut, cre]
Maintainer Koen Plevoets <koen.plevoets@ugent.be>
NeedsCompilation no
Repository CRAN
Date/Publication 2016-01-13 17:26:11

R topics documented:

svs-package .................................................. 2
cd_plot ......................................................... 4
centers_ca ................................................... 5
complete_pvpick ............................................. 6
Ctxt_Dut.txt ................................................. 7
Ctxt_Eng.txt ................................................ 7
Ctxt_Fra.txt ................................................ 8
dist_chisquare ............................................. 8
dist_cosine .................................................. 9
Description

This package offers various tools for semantic vector spaces. There are techniques for correspondence analysis (simple, multiple and discriminant), latent semantic analysis, probabilistic latent semantic analysis, non-negative matrix factorization, latent class analysis and EM clustering. Furthermore, the package has specialized distance measures and plotting functions as well as some helper functions.

Contents

This package contains the following raw data files (in the folder extdata):

- **SndT_Fra.txt** Seventeen Dutch source words and their French translations.
- **SndT_Eng.txt** Seventeen Dutch source words and their English translations.
- **InvT_Fra.txt** Seventeen Dutch target words and their French source words.
- **InvT_Eng.txt** Seventeen Dutch target words and their English source words.
- **Ctxt_Dut.txt** Context words for seventeen Dutch words.
- **Ctxt_Fra.txt** Context words for seventeen Dutch words translated from French.
Context words for seventeen Dutch words translated from English.

The (fast procedures for the) techniques in this package are:

- **fast_sca** Simple correspondence analysis.
- **fast_mca** Multiple correspondence analysis.
- **fast_dca** Discriminant correspondence analysis.
- **fast_lsa** Latent semantic analysis.
- **fast_psa** Probabilistic latent semantic analysis.
- **fast_nmf** Non-negative matrix factorization.
- **fast_lca** Latent class analysis.
- **fast_E_M** EM clustering.

The complete overview of local and global weighting functions in this package can be found on `weighting_functions`.

The specialized distance measures are:

- **dist_chisquare** Chi-square distance.
- **dist_cosine** Cosine distance.
- **dist_wrt** Distance with respect to a certain point.
- **dist_wrt_centers** Distance with respect to cluster centers.

The specialized plotting functions are:

- **cd_plot** Cumulative distribution plot.
- **pc_plot** Parallel coordinate plot.

There are two helper functions for correspondence analysis:

- **freq_ca** Compute level frequencies (for a factor).
- **centers_ca** Compute coordinates for cluster centers.

There is one helper function for `pvclust`:

- **complete_pvpick** Complete the output of pvpick.

The remaining helper functions in this package are:

- **vec2ind** Transform a vector into an indicator matrix.
- **tab2dat** Transform a table into a data frame.
- **tab2ind** Transform a table into an indicator matrix.
- **outerrec** Recursive application of the outer product.
- **pmi** Pointwise mutual information.
- **MI** Mutual information.
Further reference

- Many packages contain correspondence analysis: \texttt{ca}, \texttt{FactoMineR}, \texttt{MASS} and others.
- For latent semantic analysis there is also the package \texttt{lsa}.
- The package \texttt{NMF} provides more flexibility for non-negative matrix factorization.
- For topic models there are the packages \texttt{lda} and \texttt{topicmodels}.
- Latent class analysis can also be run in the package \texttt{poLCA}.

Author

Koen Plevoets, <koen.plevoets@ugent.be>

Acknowledgements

This package has benefited greatly from the helpful comments of Lore Vandevoorde, Pauline De Baets and Gert De Sutter. Thanks to Kurt Hornik, Uwe Ligges and Brian Ripley for their valuable recommendations when proofing this package.

\begin{tabular}{ll}
\hline
\textbf{cd\_plot} & \textit{Plotting a Cumulative Distribution} \\
\hline
\end{tabular}

Description

A function for plotting a cumulative distribution.

Usage

\begin{verbatim}
cd_plot(x, inc = 0.01, col = "darkgrey", cex = 1, font = 1,
family = "", srt = -45, pch = 20, pcol = "black", pbg = "white",
pcex = cex, lcol = col, lwd = 1, lty = 1, xlim = NULL,
ylim = NULL, xlab = NULL, ylab = NULL, main = NULL, sub = NULL)
\end{verbatim}

Arguments

- \texttt{x} A numeric vector.
- \texttt{inc} The (numeric) increment for constructing the sequence from 0 to \texttt{ceiling(max(x))}, plotted on the horizontal axis.
- \texttt{col} The color of the line and the text labels: see \texttt{colors}.
- \texttt{cex} The character expansion factor: a numeric value to specify the size of the text labels.
- \texttt{font} The font of the text labels: 1 for plain, 2 for bold, 3 for italic, and 4 for bold italic.
- \texttt{family} The font family of the text labels: "serif", "sans", "mono", or one of the \texttt{Hershey} fonts.
- \texttt{srt} The rotation angle (in degrees) of the text labels.
centers_ca

| pch  | The plotting character for displaying points: see `points`. |
| pcol | The color of the plotting character: see `colors`.            |
| pbg  | The background color of the plotting character: see `colors`. |
| pceX | The character expansion factor of the plotting character: a numeric value to specify the size of the plotting character. |
| lcol | The color of the line: see `colors`.                          |
| lwd  | The line width of the line: a numeric value to specify the width of the line. |
| lty  | The line type of the line: \(0\) or "blank", \(1\) or "solid", \(2\) or "dashed", \(3\) or "dotted", \(4\) or "dotdash", \(5\) or "longdash", \(6\) or "twodash". |
| xlim | A vector of two numeric values specifying the lower and upper limit between which to plot the horizontal axis. |
| ylim | A vector of two numeric values specifying the lower and upper limit between which to plot the vertical axis. |
| xlab | A character string for labelling of the horizontal axis.      |
| ylab | A character string for labelling of the vertical axis.        |
| main | A character string for the main title of the plot.            |
| sub  | A character string for the subtitle of the plot.              |

Value

A cumulative distribution plot.

Examples

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
               header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.SndT_Fra <- fast_sca(SndT_Fra)
dis.SndT_Fra <- dist_wrt(sca.SndT_Fra$pos1)
cd_plot(dis.SndT_Fra)
```

---

### centers_ca

#### Compute Coordinates for Cluster Centers

**Description**

A helper function for computing the coordinates of cluster centers (typically used in correspondence analysis).

**Usage**

```r
centers_ca(x, clusters, freq)
```
Arguments

- **x**: A numeric matrix.
- **clusters**: A clustering of the row levels of `x`: either a list or the output of `kmeans`.
- **freq**: An optional vector of frequency counts for the row levels of `x`.

Value

A matrix containing the coordinates of the cluster centers.

See Also

`freq_ca`.

Examples

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.SndT_Fra <- fast_sca(SndT_Fra)
kcl.SndT_Fra <- kmeans(sca.SndT_Fra$pos1, centers = 7)
centers_ca(sca.SndT_Fra$pos1, clusters = kcl.SndT_Fra, freq = freq_ca(SndT_Fra[, 1]))
```

---

**complete_pvpick**  
*Complete the Output of pvpick*

Description

A helper function to add the missing singleton clusters in the output of `pvpick` (from the package `pvclust`).

Usage

`complete_pvpick(clusters, labels)`

Arguments

- **clusters**: A clustering by a call to `pvpick`.
- **labels**: A character vector containing the exhaustive set of levels.

Value

A list with the singleton clusters inserted at the end (so that the set of clusters is exhaustive).
**Context Words for seventeen Dutch Words**

**Description**

The frequency table of seventeen Dutch synonyms of *beginnen* ("to begin") and their context words (from the Dutch Parallel Corpus).

**Format**

A table with 17 rows and 1404 columns.

**Examples**

```r
Ctx_Dut <- read.table(system.file("extdata", "Ctx_Dut.txt", package = "svs"),
  header = TRUE, sep = \"\t\", quote = \"\"", encoding = "UTF-8")
sca.Ctx_Dut <- fast_sca(data.matrix(Ctx_Dut))
sca.Ctx_Dut
lsa.Ctx_Dut <- fast_lsa(data.matrix(Ctx_Dut))
lsa.Ctx_Dut
```

**Context Words for seventeen Dutch Words Translated from French**

**Description**

The frequency table of seventeen Dutch synonyms of *beginnen* ("to begin") and their context words in texts translated from English (from the Dutch Parallel Corpus).

**Format**

A table with 17 rows and 609 columns.

**Examples**

```r
Ctx_Eng <- read.table(system.file("extdata", "Ctx_Eng.txt", package = "svs"),
  header = TRUE, sep = \"\t\", quote = \"\"", encoding = "UTF-8")
sca.Ctx_Eng <- fast_sca(data.matrix(Ctx_Eng))
sca.Ctx_Eng
lsa.Ctx_Eng <- fast_lsa(data.matrix(Ctx_Eng))
lsa.Ctx_Eng
```
Context Words for seventeen Dutch Words Translated from French

Description
The frequency table of seventeen Dutch synonyms of *beginnen* ("to begin") and their context words in texts translated from French (from the Dutch Parallel Corpus).

Format
A table with 17 rows and 612 columns.

Examples
```r
cxt_fra <- read.table(system.file("extdata", "Ctxt_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.Ctxt_Fra <- fast_sca(data.matrix(Ctxt_Fra))
lsa.Ctxt_Fra <- fast_lsa(data.matrix(Ctxt_Fra))
```

dist_chisquare
Compute Chi-square Distances

Description
A function for computing chi-square distances.

Usage
```r
dist_chisquare(x, diag = FALSE, upper = FALSE)
dist_chisq(x, diag = FALSE, upper = FALSE)
```

Arguments
- `x` A numeric matrix (containing coordinates).
- `diag` Logical specifying whether the diagonal of the resulting distance matrix should be printed.
- `upper` Logical specifying whether the upper triangle of the resulting distance matrix should be printed.

Value
A distance matrix.
dist_cosine

Examples
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
tab.SndT_Fra <- table(SndT_Fra)
dist_chisquare(tab.SndT_Fra)

dist_cosine

Compute Cosine Distances

Description
A function for computing cosine distances.

Usage
dist_cosine(x, diag = FALSE, upper = FALSE)
dist_cosine(x, diag = FALSE, upper = FALSE)

Arguments
x A numeric matrix (containing coordinates).
diag Logical specifying whether the diagonal of the resulting distance matrix should
  be printed.
upper Logical specifying whether the upper triangle of the resulting distance matrix
  should be printed.

Details
The cosine distance equals 1 - the cosine similarity.

Value
A distance matrix.

Examples
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
lsa.SndT_Fra <- fast_lsa(SndT_Fra)
dist_cosine(lsa.SndT_Fra$pos[, 1:7])
**dist_wrt**  
*Compute Distances with respect to a certain Point*

**Description**
A function for computing (euclidean) distances with respect to a certain specified point.

**Usage**
```
dist_wrt(x, wrt = NULL)
```

**Arguments**
- `x` A numeric matrix (containing coordinates).
- `wrt` A specification of the point with respect to which to compute all distances: can be either a vector or the character label of one of the row levels in `x`. If `NULL` or `NA`, then the origin (i.e. the point `c(0, 0, 0, ... )`) is taken as the value.

**Value**
A matrix (containing distances between the rows of `x` and `wrt`).

**Examples**
```
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
                        header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.SndT_Fra <- fast_sca(SndT_Fra)
dist_wrt(sca.SndT_Fra$pos1, wrt = "beginnen")
```

**dist_wrt_centers**  
*Compute Distances with respect to Cluster Centers*

**Description**
A function for computing (euclidean) distances with respect to specified cluster centers.

**Usage**
```
dist_wrt_centers(x, clusters, freq = NULL, members_only = TRUE)
```

**Arguments**
- `x` A numeric matrix (containing coordinates).
- `clusters` A clustering of the row levels of `x`: either a list or the output of `kmeans`.
- `freq` An optional vector of frequency counts for the row levels of `x`.
- `members_only` Logical specifying whether the distances from the cluster centers should only be computed for the cluster members.
**fast_dca**

**Value**

A list with a matrix of distances for every cluster.

**See Also**

`centers_ca`, `freq_ca`.

**Examples**

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
                        header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.SndT_Fra <- fast_sca(SndT_Fra)
kcl.SndT_Fra <- kmeans(sca.SndT_Fra$pos1, centers = 7)
dist_wrt_centers(sca.SndT_Fra$pos1, clusters = kcl.SndT_Fra, freq = freq_ca(SndT_Fra[,1]))
```

---

**Description**

A fast procedure for computing discriminant correspondence analysis.

**Usage**

```r
fast_dca(dat, clusters1 = NULL, clusters2 = NULL, members = FALSE)
```

**Arguments**

- `dat` Input data: can be a table or a data frame (but the data frame must have only two columns).
- `clusters1` A clustering of the first set of levels: either a list or the output of `kmeans`.
- `clusters2` A clustering of the second set of levels: either a list or the output of `kmeans`.
- `members` Logical indicating whether the (supplementary) coordinates for the individual levels should also be computed.

**Value**

A list with components:

- `val` The eigenvalues or principal inertias, indicating how much each latent axis explains.
- `cen1` The coordinates of the cluster centers for the first set of levels.
- `cen2` The coordinates of the cluster centers for the second set of levels.
- `mem1` If `members = TRUE`: The coordinates of the first set of individual levels.
- `mem2` If `members = TRUE`: The coordinates of the second set of individual levels.
References


Examples

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\\", encoding = "UTF-8")
sca.SndT_Fra <- fast_sca(SndT_Fra)
kcl.SndT_Fra <- kmeans(sca.SndT_Fra$pos1, centers = 7)
dca.SndT_Fra <- fast_dca(SndT_Fra, clusters1 = kcl.SndT_Fra)
dca.SndT_Fra
```

Description

A fast procedure for Expectation-Maximization clustering.

Usage

```r
fast_E_M(dat, k, tol = 1e-08)
fast_EM(dat, k, tol = 1e-08)
```

Arguments

- **dat**: Input data: can be a table or a data frame (but the data frame must have only two columns).
- **k**: Numeric specification of the number of latent classes to compute.
- **tol**: Numeric specification of the convergence criterion.

Details

This function assumes that the rows of a frequency table come from a multinomial distribution. The prior probabilities of the latent classes are initialized with a Dirichlet distribution (by means of `rdirichlet` from the package `gtools`) with `alpha` = the total frequency counts of every level.

Value

A list with components:

- **prob0**: The probabilities of the latent classes.
- **prob1**: The probabilities for the first set of levels (viz. the row levels of a frequency table). The rows of `prob1` sum to 1.
- **prob2**: The probabilities for the second set of levels (viz. the column levels of a frequency table). The rows of `prob2` sum to 1.
References


Examples

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
E_M.SndT_Fra <- fast_E_M(SndT_Fra, k = 7)
E_M.SndT_Fra
```

---

**fast_lca**  
*Latent Class Analysis*

**Description**

A fast procedure for computing latent class analysis.

**Usage**

```r
fast_lca(dat, k, tol = 1e-08, posterior = FALSE, sep = "_")
```

**Arguments**

- `dat`: Input data: can be a table or a data frame.
- `k`: Numeric specification of the number of latent classes to compute.
- `tol`: Numeric specification of the convergence criterion.
- `posterior`: Logical indicating whether the posterior probabilities of the individual observations should also be returned.
- `sep`: Character specifying the separator string for joining the levels (if `posterior = TRUE`).

**Details**

The prior probabilities of the latent classes are initialized with a Dirichlet distribution (by means of `rdirichlet` from the package `gtools`) with `alpha =` the total frequency counts of every level.

**Value**

A list with components:

- `prob`: The probabilities of the latent classes.
- `prob1-probNNN`: The probabilities for each set of levels. The columns of each `probNNN` sum to 1.
- `posterior`: If `posterior = TRUE`: An indicator matrix with the posterior probabilities of each observation.
References


Examples

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
lca.SndT_Fra <- fast_lca(SndT_Fra, k = 7)
lca.SndT_Fra
```

---

**fast_lsa**

*Latent Semantic Analysis*

**Description**

A fast procedure for computing latent semantic analysis.

**Usage**

```r
fast_lsa(dat, local_weights = "log", global_weights = "idf")
fast_lsi(dat, local_weights = "log", global_weights = "idf")
```

**Arguments**

- **dat**: Input data: can be a table or a data frame (but the data frame must have only two columns).
- **local_weights**: Character specification of the local weighting function (without a prefix): see Weighting functions.
- **global_weights**: Character specification of the global weighting function (without a prefix): see Weighting functions.

**Value**

A list with components:

- **val**: The singular values, indicating how much each latent axis explains.
- **pos1**: The coordinates of the first set of levels (*viz.* the row levels of a frequency table).
- **pos2**: The coordinates of the second set of levels (*viz.* the column levels of a frequency table).

**References**


Multiple Correspondence Analysis

**Description**

A fast procedure for computing multiple correspondence analysis.

**Usage**

```r
fast_mca(dat, nfac = FALSE)
```

**Arguments**

- `dat` : Input data: has to be a data frame (with any number of columns).
- `nfac` : Logical indicating whether the number of factors (i.e. the number of columns in `dat`) is a divisor for the eigenvalues (principal inertias) and the coordinates.

**Value**

A list with components:

- `val` : The eigenvalues or principal inertias, indicating how much each latent axis explains.
- `pos` : The coordinates of all levels.

**References**


**Examples**

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
                       header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
mca.SndT_Fra <- fast_mca(SndT_Fra)
mca.SndT_Fra
```
fast_nmf

Non-negative Matrix Factorization

Description
A fast procedure for non-negative matrix factorization.

Usage
fast_nmf(dat, k, type = "KL", tol = 1e-08)
fast_nmf_KL(dat, k, tol = 1e-08)
fast_nmf_Fr(dat, k, tol = 1e-08)
fast_nmf_Al(dat, k, tol = 1e-08)

Arguments
- dat: Input data: can be a table or a data frame (but the data frame must have only two columns).
- k: Numeric specification of the number of latent axes to compute.
- type: Character specification of the type of optimization: can in the current implementation be either "KL" for the Kullback-Leibler divergence, "Frobenius" or "euclidean" (or abbreviations thereof) for the euclidean distance, or "ALS" for alternating least squares.
- tol: Numeric specification of the convergence criterion.

Value
A list with components:
- pos1: The coordinates of the first set of levels (viz. the row levels of a frequency table).
- pos2: The coordinates of the second set of levels (viz. the column levels of a frequency table).

References
fast_psa

Examples

 SNDT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
   header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
nmf.SndT_Fra <- fast_nmf(SndT_Fra, k = 7)
nmf.SndT_Fra

Description

A fast procedure for computing probabilistic latent semantic analysis.

Usage

fast_psa(dat, k, symmetric = FALSE, tol = 1e-08)
fast_psi(dat, k, symmetric = FALSE, tol = 1e-08)
fast_plsa(dat, k, symmetric = FALSE, tol = 1e-08)
fast_plsi(dat, k, symmetric = FALSE, tol = 1e-08)

Arguments

dat Input data: can be a table or a data frame (but the data frame must have only two columns).
k Numeric specification of the number of latent classes to compute.
symmetric Logical indicating whether to compute the symmetric or the asymmetric solution.
tol Numeric specification of the convergence criterion.

Details

From version 1.1.0 of the svs package on, probabilistic latent semantic analysis is a special case of latent class analysis.

Value

A list with components:

prob0 The probabilities of the latent classes.
prob1 The probabilities for the first set of levels (viz. the row levels of a frequency table). The rows of prob1 sum to 1 if symmetric = FALSE, the columns sum to 1 if symmetric = TRUE.
prob2 The probabilities for the second set of levels (viz. the column levels of a frequency table). The columns of prob2 sum to 1.
References

Examples
```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
psa.SndT_Fra <- fast_psa(SndT_Fra, k = 7)
psa.SndT_Fra
```

---

**fast_sca**

**Simple Correspondence Analysis**

Description
A fast procedure for computing simple correspondence analysis.

Usage
```r
fast_sca(dat)
```

Arguments
- `dat`: Input data: can be a table or a data frame (but the data frame must have only two columns).

Value
A list with components:
- `val`: The eigenvalues or principal inertias, indicating how much each latent axis explains.
- `pos1`: The coordinates of the first set of levels (viz. the row levels of a frequency table).
- `pos2`: The coordinates of the second set of levels (viz. the column levels of a frequency table).

References

Examples
```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.SndT_Fra <- fast_sca(SndT_Fra)
sca.SndT_Fra
```
freq_ca

**Compute Level Frequencies (for a Factor)**

**Description**

A helper function for computing the frequency of each factor level (typically used in correspondence analysis).

**Usage**

```
freq_ca(dat, nfac = FALSE)
```

**Arguments**

- `dat` A factor or a data frame.
- `nfac` Logical indicating whether the number of factors (i.e. the number of columns in `dat`) is a divisor for the level frequencies.

**Value**

A vector containing the frequency counts of every level.

**Examples**

```
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
                       header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
freq_ca(SndT_Fra)
```

---

**InvT_Eng.txt**

**Seventeen Dutch Target Words and their English Source Words**

**Description**

The occurrences of seventeen Dutch synonyms of *beginnen* ("to begin") and their English source words (from the Dutch Parallel Corpus).

**Format**

A data frame with 782 rows and 2 variables.

- `source_Eng` The English source word.
- `target_Dut` The Dutch target word.
Examples

```r
InvT_Eng <- read.table(system.file("extdata", "InvT_Eng.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.InvT_Eng <- fast_sca(InvT_Eng)
sca.InvT_Eng
lsa.InvT_Eng <- fast_lsa(InvT_Eng)
lsa.InvT_Eng
```

---

**InvT_Fra.txt**  **Seventeen Dutch Target Words and their French Source Words**

Description

The occurrences of seventeen Dutch synonyms of *beginnen* ("to begin") and their French source words (from the Dutch Parallel Corpus).

Format

A data frame with 856 rows and 2 variables.

- **source_Fra** The French source word.
- **target_Dut** The Dutch target word.

Examples

```r
InvT_Fra <- read.table(system.file("extdata", "InvT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.InvT_Fra <- fast_sca(InvT_Fra)
sca.InvT_Fra
lsa.InvT_Fra <- fast_lsa(InvT_Fra)
lsa.InvT_Fra
```

---

**MI**  **Mutual Information**

Description

A function for computing the mutual information.

Usage

```r
MI(x, base = 2)
mi(x, base = 2)
```
Arguments

- `x` A table (i.e. an object which can be converted to an array).
- `base` Numeric specification of the base with respect to which logarithms are computed.

Value

A numeric value containing the mutual information.

See Also

`pmi`.

Examples

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
tab.SndT_Fra <- table(SndT_Fra)
MI(tab.SndT_Fra)
```

Description

A helper function for computing the outer product of two or more arrays.

Usage

```r
outerec(...)```

Arguments

- `...` The specification of two or more arrays (separated by comma’s or contained in a list).

Value

An array with the outer product of all the arrays specified in `...`. 

---

**Recursive Application of the Outer Product**

Description

A helper function for computing the outer product of two or more arrays.

Usage

`outerec(...)`

Arguments

- `...` The specification of two or more arrays (separated by comma’s or contained in a list).

Value

An array with the outer product of all the arrays specified in `...`. 

---
pc_plot

Plotting Parallel Coordinates

Description

A function for plotting parallel coordinates.

Usage

```r
pc_plot(x, col = "darkgrey", cex = 1, font = 1, family = "", pch = 20,
        pcol = col, pcex = cex, lcol = col, lwd = 1, lty = 1,
        acol = "black", alwd = 1, alty = 1, las = 1, add_scale = FALSE,
        main = NULL, sub = NULL)
```

Arguments

- `x`: A numeric matrix.
- `col`: The color of the text labels, points and connecting lines: see `colors`.
- `cex`: The character expansion factor: A numeric value to specify the size of the text labels and the points.
- `font`: The font of the text labels: 1 for plain, 2 for bold, 3 for italic, and 4 for bold italic.
- `family`: The font family of the text labels: "serif", "sans", "mono", or one of the Hershey fonts.
- `pch`: The plotting character for displaying points: see `points`.
- `pcol`: The color of the plotting character: see `colors`.
- `pcex`: The character expansion factor of the plotting character: a numeric value to specify the size of the plotting character.
- `lcol`: The color of the connecting lines: see `colors`.
- `lwd`: The line width of the connecting lines: a numeric value to specify the width of the connecting lines.
- `lty`: The line type of the connecting lines: 0 or "blank", 1 or "solid", 2 or "dashed", 3 or "dotted", 4 or "dotdash", 5 or "longdash", 6 or "twodash".
- `acol`: The color of the parallel axes: see `colors`.
- `alwd`: The line width of the parallel axes: a numeric value to specify the width of the parallel axes.
- `alty`: The line type of the parallel axes: 0 or "blank", 1 or "solid", 2 or "dashed", 3 or "dotted", 4 or "dotdash", 5 or "longdash", 6 or "twodash".
- `las`: The reading direction of the labels on the axes ("label axis style"): either a numeric value between 0 and 3 (see las in `par`), or a character value matching either "horizontal" or "vertical".
- `add_scale`: Logical specifying whether to add a scale for the parallel axes (which are normalized).
- `main`: A character string for the main title of the plot.
- `sub`: A character string for the subtitle of the plot.
Pointwise Mutual Information

pmi

Description
A function for computing the pointwise mutual information of every entry in a table.

Usage
pmi(x, normalize = FALSE, base = 2)

PMI(x, normalize = FALSE, base = 2)

Arguments
x
A table (i.e. an object which can be converted to an array).

normalize
Logical indicating whether to normalize the pointwise mutual information.

base
Numeric specification of the base with respect to which logarithms are computed.

Value
An array with the pointwise mutual information of every entry.

See Also
MI.

Examples
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
sca.SndT_Fra <- fast_sca(SndT_Fra)
pc_plot(sca.SndT_Fra$posl, las = "vertical")

tab.SndT_Fra <- table(SndT_Fra)
pmi(tab.SndT_Fra)
SndT_Eng.txt  Seventeen Dutch Source Words and their English Translations

Description
The occurrences of seventeen Dutch synonyms of *beginnen* ("to begin") and their English translations (from the Dutch Parallel Corpus).

Format
A data frame with 1117 rows and 2 variables.

- source_Dut The Dutch source word.
- target_Eng The English target word.

Examples

```r
SndT_Eng <- read.table(system.file("extdata", "SndT_Eng.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
scasndT_Eng <- fast_sca(SndT_Eng)
sca.SndT_Eng
lsa.SndT_Eng <- fast_lsa(SndT_Eng)
lsa.SndT_Eng
```

SndT_Fra.txt  Seventeen Dutch Source Words and their French Translations

Description
The occurrences of seventeen Dutch synonyms of *beginnen* ("to begin") and their French translations (from the Dutch Parallel Corpus).

Format
A data frame with 1487 rows and 2 variables.

- source_Dut The Dutch source word.
- target_Fra The French target word.

Examples

```r
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
  header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")
scasndT_Fra <- fast_sca(SndT_Fra)
sca.SndT_Fra
lsa.SndT_Fra <- fast_lsa(SndT_Fra)
lsa.SndT_Fra
```
**tab2dat**  
*Transform a Table into a Data Frame*

**Description**
A helper function for transforming a table into a data frame.

**Usage**
```r
tab2dat(tab)
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tab</code></td>
<td>A table (i.e. an object which can be converted to an array).</td>
</tr>
</tbody>
</table>

**Value**
A data frame.

---

**tab2ind**  
*Transform a Table into an Indicator Matrix*

**Description**
A helper function for transforming a table into an indicator matrix.

**Usage**
```r
tab2ind(tab, sep = " ", add_names = TRUE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tab</code></td>
<td>A table (i.e. an object which can be converted to an array).</td>
</tr>
<tr>
<td><code>sep</code></td>
<td>Character specifying the separator string for joining the levels.</td>
</tr>
<tr>
<td><code>add_names</code></td>
<td>Logical specifying whether to add dimnames to the resulting indicator matrix.</td>
</tr>
</tbody>
</table>

**Value**
An indicator matrix.
weighting_functions

vec2ind  Transform a Vector into an Indicator Matrix

Description
A helper function for transforming a vector into an indicator matrix.

Usage
vec2ind(x, add_names = TRUE)

Arguments
x  A vector (which will be converted to a factor).
add_names  Logical specifying whether to add dimnames to the resulting indicator matrix.

Details
This is essentially the function class.ind from the package MASS.

Value
An indicator matrix.

weighting_functions  Weighting Functions

Description
Local and global weighting functions.

Usage
lw_tf(x)
lw_raw(x)
lw_log(x)
lw_bin(x)
gw_idf(x)
gw_idf_alt(x)
Arguments

x

A numeric matrix.

Details

There are many local and global weighting functions. In this package, local weighting functions are prefixed with `lw_` and global weighting functions with `gw_`, so users can define their own weighting functions.

Local weighting functions (i.e. weighting every cell in the matrix):

- `lw_tf` Term frequency: \( f(x) = x \).
- `lw_raw` Raw frequency, which is the same as the term frequency: \( f(x) = x \).
- `lw_log` Logarithm: \( f(x) = \log(x + 1) \).
- `lw_bin` Binary: \( f(x) = 1 \) if \( x > 0 \) and \( 0 \) otherwise.

Global weighting functions, weighting the columns of the matrix (hence, these weighting functions work according to expectation for a document-term matrix, i.e. with the documents as the rows and the terms as the columns):

- `gw_idf` Inverse document frequency: \( f(x) = \log( nrow(x) / n + 1) \) where \( n = \) the number of rows in which the column >0.
- `gw_idf_alt` Alternative definition of the inverse document frequency: \( f(x) = \log( nrow(x) / n) + 1 \) where \( n = \) the number of rows in which the column >0.
- `gw_gfidf` Global frequency multiplied by inverse document frequency: \( f(x) = \text{colSums}(x) / n \) where \( n = \) the number of rows in which the column >0.
- `gw_nor` Normal(ized) frequency: \( f(x) = x / \text{colSums}(x^2) \).
- `gw_ent` Entropy: \( f(x) = 1 + \) the relative Shannon entropy.
- `gw_bin` Binary: \( f(x) = 1 \).
- `gw_raw` Raw, which is the same as binary: \( f(x) = 1 \).

Value

A numeric matrix.

See Also

`fast_lsa`. 

weighting_functions

\[
gw_gfidf(x) \\
gw_nor(x) \\
gw_ent(x) \\
gw_bin(x) \\
gw_raw(x)
\]
Examples

SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
    header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8")

tab.SndT_Fra <- table(SndT_Fra)
lw_log(tab.SndT_Fra)
gw_idf(tab.SndT_Fra)
Index

cd_plot, 3, 4
centers_ca, 3, 5, 11
colors, 4, 5, 22
complete_pvpick, 3, 6
Ctxt_Dut.txt, 2, 7
Ctxt_Eng.txt, 3, 7
Ctxt_Fra.txt, 2, 8
dist_chisq (dist_chisquare), 8
dist_chisquare, 3, 8
dist_cos (dist_cosine), 9
dist_cosine, 3, 9
dist_wrt, 3, 10
dist_wrt_centers, 3, 10
fast_dca, 3, 11
fast_E_M, 3, 12
fast_Lca (fast_E_M), 12
fast_Lca, 3, 13
fast_lsa, 3, 14, 27
fast_lsi (fast_lsa), 14
fast_mca, 3, 15
fast_nmf, 3, 16
fast_nmf_A1 (fast_nmf), 16
fast_nmf_Fr (fast_nmf), 16
fast_nmf_KL (fast_nmf), 16
fast_plsa (fast_psa), 17
fast_plsi (fast_psa), 17
fast_psa, 3, 17
fast_psi (fast_psa), 17
fast_sca, 3, 18
freq_ca, 3, 6, 11, 19
Hershey, 4, 22
InvT_Eng.txt, 2, 19
InvT_Fra.txt, 2, 20
lw_bin (weighting_functions), 26
lw_log (weighting_functions), 26
lw_raw (weighting_functions), 26
lw_tf (weighting_functions), 26
MI, 3, 20, 23
mi (MI), 20
outerec, 3, 21
par, 22
pc_plot, 3, 22
PMI (pmi), 23
pmi, 3, 21, 23
points, 5, 22
SndT_Eng.txt, 2, 24
Sndt_Fra.txt, 2, 24
svs-package, 2
tab2dat, 3, 25
tab2ind, 3, 25
vec2ind, 3, 26
Weighting functions, 14
weighting_functions, 3, 26

gw_bin (weighting_functions), 26
gw_ent (weighting_functions), 26
gw_gfidf (weighting_functions), 26
gw_idf (weighting_functions), 26
gw_idf_alt (weighting_functions), 26
gw_nor (weighting_functions), 26
gw_raw (weighting_functions), 26

weighting_functions, 14