# Package ‘svs’

June 24, 2024

**Title** Tools for Semantic Vector Spaces

**Version** 3.1.1

**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, EM clustering, logratio analysis and log-multiplicative (association) analysis. Furthermore, there are specialized distance measures, plotting functions and some helper functions.

**Depends** R (>= 4.0.0),

**Imports** gtools, graphics, stats, methods, Matrix, utils

**Suggests** igraph, MASS, pvclust

**License** GPL-3

**Encoding** UTF-8

**Date** 2024-06-24

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** Koen Plevoets [aut, cre] (<https://orcid.org/0000-0003-3889-1809>)

**Maintainer** Koen Plevoets <koen.plevoets@ugent.be>

**Repository** CRAN

**Date/Publication** 2024-06-24 13:10:01 UTC

## Contents

---

`svs-package` *Tools for Semantic Vector Spaces*

---

## 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, EM clustering, logratio analysis and log-multiplicative (association) analysis. 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.

`Ctxt_Eng.txt` 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.

`fast_lra` Logratio analysis.

`fast_lma` Log-multiplicative (association) analysis.

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`.

There is one helper function for **igraph**:

`layout4bipartite` Create a layout matrix for a bipartite graph.

The remaining helper functions in this package are:

`rep4dat` Repeat the rows of a data frame according to a frequency column.

`vec2ddc` Transform a vector into a double-coded matrix.

`dat2ddc` Transform a data frame into a double-coded matrix.

`vec2ind` Transform a vector into an indicator matrix.

`tab2dat` Transform a table into a data frame.

`tab2ind` Transform a table into an indicator matrix.

`dat2ind` Transform a data frame into an indicator matrix.

`outerec` Recursive application of the outer product.

`pmi` Pointwise mutual information.

`MI` Mutual information.

`log_or_0` Logarithmic transform.

### Further reference

- Many packages contain correspondence analysis: **ca**, **FactoMineR**, **MASS** and others.
- For latent semantic analysis there is also the package **lsa**.
- The package **NMF** provides more flexibility for non-negative matrix factorization.
- For topic models there are the packages **lda** and **topicmodels**.
- Latent class analysis can also be run in the package **poLCA**.
- For log-ratio analysis there is also the package **easyCODA**.
- The package **gnm** offers much flexibility for association analysis, i.e. fitting log-multiplicative or Goodman's RC models.

### Link

As from 2023, this package is part of Module 10: Multivariate data analysis with R of the Summer School *Methods in Language Sciences*.

### 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.

## `cd_plot` *Plotting a Cumulative Distribution*

### Description

A function for plotting a cumulative distribution.

### Usage

```
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
)
```

### Arguments

`x` A numeric vector.

`inc` The (numeric) increment for constructing the sequence from 0 to `ceiling(max(x))`, plotted on the horizontal axis.

`col` The color of the line and the text labels: see `colors`.

`cex` The character expansion factor: a numberic value to specify the size of the text labels.

`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.

`srt` The rotation angle (in degrees) of the text labels.

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

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

```
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

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

## Ctxt_Dut.txt *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

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

## Ctxt_Eng.txt *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

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

## `Ctxt_Fra.txt` *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

```
Ctxt_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))
sca_Ctxt_Fra
lsa_Ctxt_Fra <- fast_lsa(data.matrix(Ctxt_Fra))
lsa_Ctxt_Fra
```

## `dat2ddc` *Transform a Data Frame into a Double-Coded Matrix*

### Description

A helper function for transforming all columns of a data frame into a double-coded matrix.

### Usage

```
dat2ddc(dat, limits = NULL)

dat2dc(dat, limits = NULL)
```

### Arguments

| | |
|---|---|
| `dat` | A data frame containing only numeric vectors or ordered factors as columns. |
| `limits` | A vector, matrix or list specifying the lower and upper limits for the columns in `dat`. A single vector of length two applies the same limits to all columns. The matrix or list must either have the same number of columns/components as the number of the number of columns in `dat` or they must have names. The names can be a subset of the column names of `dat`. In any case, the limits for each column must be a vector of length two, i.e. the matrix must have two rows or each list component should be a vector of length two. If left unspecified (the default), then the minimum and maximum of each column in `dat` will be taken as (separate) limits. |

### Value

A matrix with doubly-coded columns.

### See Also

vec2ddc.

## dat2ind *Transform a Data Frame into an Indicator Matrix*

### Description

A helper function for transforming a data frame into an indicator matrix.

### Usage

```
dat2ind(dat, add_names = TRUE)
```

### Arguments

| | |
|---|---|
| dat | A data frame. |
| add_names | Logical specifying whether to add rownames to the resulting indicator matrix. |

### Value

An indicator matrix.

### See Also

vec2ind.

## dist_chisquare *Compute Chi-square Distances*

### Description

A function for computing chi-square distances.

### Usage

```
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.

### Examples

```
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
   header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8",
   stringsAsFactors = FALSE)
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_cos(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",
   stringsAsFactors = FALSE)
lsa_SndT_Fra <- fast_lsa(SndT_Fra)
dist_cosine(lsa_SndT_Fra$pos1[, 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",
   stringsAsFactors = FALSE)
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.

### Value

A list with a matrix of distances for every cluster.

### See Also

`centers_ca`, `freq_ca`.

### Examples

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

## fast_dca *Discriminant Correspondence Analysis*

### Description

A fast procedure for computing discriminant correspondence analysis.

### Usage

```
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

Abdi, H. (2007) Discriminant correspondence analysis. In: N. Salkind (ed.) *Encyclopedia of measurement and statistics*. Thousand Oaks: SAGE.

### Examples

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

## `fast_E_M` *EM clustering*

### Description

A fast procedure for Expectation-Maximization clustering.

### Usage

```
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

Dempster, A. P., N. M. Laird and D. B. Rubin (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society, series B* **39** (1), 1–38.

### Examples

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

```
fast_lca(dat, k, tol = 1e-08, posterior = FALSE, transform = 1, 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. |
| `transform` | Numeric specification of the power transformation to be applied on the posterior distribution; see `fast_psa`. |
| `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:

| | |
|---|---|
| `prob0` | The probabilities of the latent classes. |
| `prob1-prob...` | The probabilities for each set of levels. The columns of each `prob...` sum to 1. |
| `posterior` | If `posterior = TRUE`: An indicator matrix with the posterior probabilities of each observation. |

### References

Agresti, A. (2013) *Categorical data analysis*. Hoboken: John Wiley and Sons, 535–542.

### Examples

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

# `fast_lma` *Log-Multiplicative Association Analysis*

## Description

A fast procedure for computing log-multiplicative analysis, i.e. Goodman's _RC(M)_ association model.

## Usage

```
fast_lma(
  dat,
  k,
  weights = "marginal",
  tol = 1e-08,
  base = exp(1),
  init = "marginal"
)

fast_rca(
  dat,
  k,
  weights = "marginal",
  tol = 1e-08,
  base = exp(1),
  init = "marginal"
)
```

## Arguments

`dat` Input data: can be a table or a data frame.

`k` Numeric specification of the number of latent axes to compute (i.e. k = M).

`weights` Character specification of the weights applied to standardize the coordinates: can be one of `"marginal"`, `"uniform"`, `"unit"` or `"none"`.

`tol` Numeric specification of the convergence criterion.

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

`init` Character specification of the initialization scheme for the marginal parameters: can be either `"kateri"` or `"marginal"`. This argument may change in future versions of the **svs** package.

## Details

For now (i.e. version 3.0.0 of the **svs** package), the data frame must have only two columns.

### Value

A list with components:

`mar` A list with marginal parameters in components `mar1` and `mar2`; not so important for the analysis.

`val` The association parameters, indicating how much association 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

Goodman, L. A. (1979) Simple models for the analysis of association in cross-classifications having ordered categories. *Journal of the American statistical association* **74** (367), 537–552.

Kateri, M. (2014) *Contingency table analysis. Methods and implementation using R*. New York: Springer-Birkhauser.

Wong, R. S.-K. (2010) *Association models*. Thousand Oaks: SAGE.

### Examples

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

## `fast_lra` *Logratio Analysis*

### Description

A fast procedure for computing logratio analysis.

### Usage

```
fast_lra(dat, base = exp(1))
```

### Arguments

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

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

### Value

A list with components:

`val` The eigenvalues (i.e. squared 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

Greenacre, M. (2019) *Compositional data analysis in practice*. Boca Raton: Chapman and Hall/CRC.

Van den Boogaart, K. G. and R. Tolosana-Delgado (2013) *Analyzing compositional data with R*. Berlin: Springer.

### Examples

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

## `fast_lsa` *Latent Semantic Analysis*

### Description

A fast procedure for computing latent semantic analysis.

### Usage

```
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

Deerwester, S., S. T. Dumais, G. W. Furnas, Th. K. Landauer and R. Harshman (1990) Indexing by latent semantic analysis. *Journal of the American society for information science* **41** (6), 391–407.

Landauer, Th. K. and S. T. Dumais (1997) A solution to Plato's problem: the latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological review* **104**, 211–240.

### Examples

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

## fast_mca *Multiple Correspondence Analysis*

### Description

A fast procedure for computing multiple correspondence analysis.

### Usage

```
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

Greenacre, M. (2017) *Correspondence analysis in practice, Third edition.* Boca Raton: Chapman and Hall/CRC.

### Examples

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

Lee, D. D. and H. S. Seung (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* **401**, 788–791.

Lee, D. D. and H. S. Seung (2001) Algorithms for non-negative matrix factorization. *Advances in neural information processing systems* **13**, 556–562.

### Examples

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

## fast_psa *Probabilistic Latent Semantic Analysis*

### Description

A fast procedure for computing probabilistic latent semantic analysis.

### Usage

```
fast_psa(dat, k, symmetric = FALSE, transform = 1, tol = 1e-08)

fast_psi(dat, k, symmetric = FALSE, transform = 1, tol = 1e-08)

fast_plsa(dat, k, symmetric = FALSE, transform = 1, tol = 1e-08)

fast_plsi(dat, k, symmetric = FALSE, transform = 1, 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.

`transform` Numeric specification of the "tempering" transformation as explained in Hofmann (1999: 51-52).

`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

Hofmann, Th. (1999). Probabilistic latent semantic indexing. *SIGIR'99: Proceedings of the 22nd annual international SIGIR conference on research and development in information retrieval*, 50–57.

## Examples

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

```
fast_sca(dat, transform = 1)
```

## Arguments

| | |
|---|---|
| `dat` | Input data: can be a table or a data frame (but the data frame must have only two columns). |
| `transform` | Numeric specification of the power transformation to be applied on the data. |

### 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

Greenacre, M. (2017) *Correspondence analysis in practice, Third edition.* Boca Raton: Chapman and Hall/CRC.

### Examples

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

## freq_ca *Compute Level Frequencies (for a Factor or Vector)*

### 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, (character) vector 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",
   stringsAsFactors = FALSE)
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

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

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

## layout4bipartite *Create a Layout Matrix for a Bipartite Graph*

### Description

A helper function for creating a layout matrix for a bipartite graph, based on the dimensions of its biadjacency matrix.

### Usage

```
layout4bipartite(nrow, ncol, rowsLeft = TRUE)

layout4diagram(nrow, ncol, rowsLeft = TRUE)
```

### Arguments

| | |
|---|---|
| `nrow` | Numeric specifying the number of rows of the biadjacency matrix. |
| `ncol` | Numeric specifying the number of columns of the biadjacency matrix. |
| `rowsLeft` | Logical specifying whether the rows of the biadjacency matrix should appear to the left of the columns. |

### Details

This is essentially an alternative for `layout_as_bipartite()` from the **igraph** package. Bipartite graphs are typically represented by a biadjacency matrix, the dimensions of which can be used for constructing the layout. In addition, the coordinates for the vertices are somewhat different from those in **igraph**.

### Value

A matrix.

## log_or_0 *Logarithmic transform*

### Description

A function for computing the logarithm of every entry in a table with 0 for zero entries.

### Usage

```
log_or_0(x, base = exp(1))
```

### Arguments

| | |
|---|---|
| x | A table or a (sparse) matrix. |
| base | Numeric specification of the base with respect to which logarithms are computed. |

### Value

An array with the logarithm of every entry and 0 for all zero entries.

### Examples

```
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
   header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8",
   stringsAsFactors = FALSE)
tab_SndT_Fra <- table(SndT_Fra)
log_or_0(tab_SndT_Fra)
```

## MI *Mutual Information*

### Description

A function for computing the mutual information.

### Usage

```
MI(x, base = 2)

mi(x, base = 2)
```

### Arguments

| | |
|---|---|
| x | A table or a (sparse) matrix. |
| 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

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

## `outerec` *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

```
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.

### Value

A parallel coordinate plot.

### Examples

```
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
   header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8",
   stringsAsFactors = FALSE)
sca_SndT_Fra <- fast_sca(SndT_Fra)
pc_plot(sca_SndT_Fra$pos1, las = "vertical")
```

## pmi *Pointwise Mutual Information*

### 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 or a (sparse) matrix.

`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",
   stringsAsFactors = FALSE)
tab_SndT_Fra <- table(SndT_Fra)
pmi(tab_SndT_Fra)
```

---

## `rep4dat` *Repeat the Rows of a Data Frame according to a Frequency Column*

---

### Description

A helper function for repeating the rows a data frame according to a frequency column.

### Usage

```
rep4dat(dat, freqName = "Freq")
```

### Arguments

`dat` A data frame.

`freqName` Character specifying the name of the frequency column.

### Value

A data frame.

### Examples

```
SndT_Fra <- read.table(system.file("extdata", "SndT_Fra.txt", package = "svs"),
   header = TRUE, sep = "\t", quote = "\"", encoding = "UTF-8",
   stringsAsFactors = FALSE)
tab_SndT_Fra <- table(SndT_Fra)
dat_SndT_Fra <- as.data.frame(tab_SndT_Fra)
rep4dat(dat_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

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

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

```
tab2dat(tab)
```

## Arguments

| | |
|---|---|
| tab | A table or (sparse) matrix. |

## 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

```
tab2ind(tab, add_names = TRUE)
```

## Arguments

| | |
|---|---|
| tab | A table or (sparse) matrix. |
| add_names | Logical specifying whether to add rownames to the resulting indicator matrix. |

## Value

An indicator matrix.

## vec2ddc *Transform a Vector into a Double-Coded Matrix*

### Description

A helper function for transforming a vector into a double-coded matrix.

### Usage

```
vec2ddc(x, limits = c(NA, NA))

vec2dc(x, limits = c(NA, NA))
```

### Arguments

| | |
|---|---|
| `x` | A vector, either numeric or an ordered factor. |
| `limits` | Numeric vector of length two specifying the limits (or "poles") of the underlying ordinal variable. If left unspecified (the default), then the minimum and maximum in `x` will be taken. |

### Details

Double coding, also called data doubling, is a way of indicating how the values of an ordinal variable relate to the lower and upper limits of the variable. Columns `-` and `+` express how close each (observed) value is to these respective limits.

### Value

A matrix with column names `-` and `+`.

### References

Greenacre, M. (2014) Data doubling and fuzzy coding. In: J. Blasius and M. Greenacre (eds.) *Visualization and verbalization of data*. Chapman and Hall/CRC, 239–253. Greenacre, M. (2017) *Correspondence analysis in practice, Third edition*. Boca Raton: Chapman and Hall/CRC, 201–208.

### See Also

`dat2ddc`.

## 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 internally be converted to a factor). |
| `add_names` | Logical specifying whether to add dimnames to the resulting indicator matrix. |

### Details

As of version 2.0.x of the `svs` package, this is essentially a wrapper for `t(fac2sparse())` from the **Matrix** package.

### Value

An indicator matrix.

### See Also

`dat2ind`.

## 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)

gw_gfidf(x)

gw_nor(x)

gw_ent(x)

gw_bin(x)

gw_raw(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) = colSums(x) / n* where *n* = the number of rows in which the column >*0*.

`gw_nor` Normal(ized) frequency: *f(x) = x / 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.

## Examples

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

# Index