Package ‘RMaCzek’

May 7, 2026

Type Package

Title Czekanowski's Diagrams

Version 1.6.1

Date 2025-12-17

Encoding UTF-8

Maintainer Krzysztof Bartoszek `<krzbar@protonmail.ch>`

Description Allows for production of Czekanowski's Diagrams with clusters. See K. Bartoszek, A. Vasterlund (2020) <doi:10.2478/bile-2020-0008> and K. Bartoszek, Y. Luo (2023) <doi:10.14708/ma.v51i2.7259>. The suggested 'FuzzyDBScan' package (which allows for fuzzy clustering) can be obtained from <`https://github.com/henrifnk/FuzzyDBScan/`> (or from CRAN's Archive <`https://cran.r-project.org/src/contrib/Archive/FuzzyDBScan/`>).

Depends R(>= 3.6)

Imports GA(>= 3.2), graphics, methods, seriation(>= 1.3.4), stats, e1071, ecp, RColorBrewer, utils

Suggests FuzzyDBScan

License GPL-3

Collate czek_matrix.R dot_functions.R plot.czek_matrix.R register_seriate_ga.R RMaCzek.R seriate_ga.R Um_factor.R utils_czek_matrix.R cluster_distance.R data.R read_maczek.R

LazyLoad yes

LazyData true

NeedsCompilation no

Author Albin Vasterlund [aut],
Krzysztof Bartoszek [cre, aut, ths],
Ying Luo [aut],
Piotr Jaskulski [ctb]

RoxygenNote 7.3.1

Repository CRAN

Date/Publication 2025-12-22 09:20:02 UTC

Contents

`cluster_dist` *Calculate the distance matrix between clusters.*

Description

Calculate the distance matrix for a czek_matrix with clustering result or a data set with its clustering labels.

Usage

```
cluster_dist(x, y, distfun = dist, dist_method = "average")
```

Arguments

`x`	A data set or a matrix with class czek_matrix.
`y`	If x is the data set, y is the cluster label.
`distfun`	Specifies which distance function should be used.
`dist_method`	Four linkage criteria: single, complete, average and SSD.

Value

A distance matrix.

Examples

```
# Clustering Result on czek_matrix
x = czek_matrix(iris[,-5], cluster = TRUE, num_cluster = 3)
dist_czek = cluster_dist(x)
plot(czek_matrix(dist_czek))

# Clustering Result on a Data Set with Clustering Labels
dist_data = cluster_dist(x = iris[,-5], y = iris$Species)
plot(czek_matrix(dist_data))
```

`czek_matrix` *Preprocess data to produce Czekanowski's Diagram.*

Description

Preprocess the data to generate a matrix of category czek_matrix for generating Czekanowski's Diagram. This method also offers exact and fuzzy clustering algorithms for Czekanowski's Diagram.

Usage

```
czek_matrix(
  x,
  order = "OLO",
  n_classes = 5,
  interval_breaks = NULL,
  monitor = FALSE,
  distfun = dist,
  scale_data = TRUE,
  focal_obj = NULL,
  as_dist = FALSE,
  original_diagram = FALSE,
  column_order_stat_grouping = NULL,
  dist_args = list(),
  cluster = FALSE,
  cluster_type = "exact",
  num_cluster = 3,
  sig.lvl = 0.05,
  scale_bandwidth = 0.05,
  min.size = 30,
  eps = 0.01,
  pts = c(1, 5),
  alpha = 0.2,
  theta = 0.9,
  ...
)
```

Arguments

`x`	A numeric matrix, data frame or a 'dist' object.
`order`	Specifies which seriation method should be applied. The standard setting is the seriation method OLO. If NA or NULL, then no seriation is done and the original ordering is saved. The user may provide their own ordering, through a number vector of indices. Also in this case no rearrangement will be done.
`n_classes`	Specifies how many classes the distances should be divided into. The standard setting is 5 classes.
`interval_breaks`	Specifies the partition boundaries for the distances. As a standard setting, each class represents an equal amount of distances. If the interval breaks are positive and sum up to 1, then it is assumed that they specify percentages of the distances in each interval. Otherwise, if provided as a numeric vector not summing up to 1, they specify the exact boundaries for the symbols representing distance groups.
`monitor`	Specifies if the distribution of the distances should be visualized. The standard setting is that the distribution will not be visualized. TRUE and "cumulativ_plot" is available.
`distfun`	Specifies which distance function should be used. Standard setting is the dist function which uses the Euclidean distance. The first argument of the function has to be the matrix or data frame containing the data.
`scale_data`	Specifies if the data set should be scaled. The standard setting is that the data will be scaled.
`focal_obj`	Numbers or names of objects (rows if x is a dataset and not 'dist' object) that are not to take part in the reordering procedure. These observations will be placed as last rows and columns of the output matrix. See Details.
`as_dist`	If TRUE, then the distance matrix of x is returned, with object ordering, instead of the matrix with the levels assigned in place of the original distances.
`original_diagram`	If TRUE, then the returned matrix corresponds as close as possible to the original method proposed by Czekanowski (1909). The levels are column specific and not matrix specific. See Details
`column_order_stat_grouping`	If original_diagram is TRUE, then here one can pass the partition boundaries for the ranking in each column.
`dist_args`	Specifies further parameters that can be passed on to the distance function.
`cluster`	If TRUE, Czekanowski's clustering is performed.
`cluster_type`	Specifies the cluster type and it can be 'exact' or 'fuzzy'.
`num_cluster`	Specifies the number of clusters.
`sig.lvl`	The threshold for testing a change point is statistically significant. This value is passed to ecp::e.divisive().
`scale_bandwidth`	A ratio to control the width of the reaching range.
`min.size`	Minimum number of observations between change points.

`eps`	A vector of epsilon values for FDBScan.
`pts`	A vector of minimum points for FDBScan.
`alpha`	The weighting factor for density score adjustments.
`theta`	The weighting factor for density score adjustments.
`...`	Further parameters that can be passed on to the seriate function in the seriation package.

Value

The function returns a matrix with class czek_matrix. The returned object is expected to be passed to the plot function if as_dist is FALSE. If as_dist is passed as TRUE, then a czek_matrix object is returned that is not suitable for plotting. As an attribute of the output the optimized criterion value is returned. However, this is a guess based on seriation::seriate()'s and seriation::criterion()'s manuals. If something else was optimized, e.g. due to user's parameters, then this will be wrong. If unable to guess, then NA saved in the attribute.

Examples

```
# Set data ####
x<-mtcars

# Different type of input that give same result ############
czek_matrix(x)
czek_matrix(stats::dist(scale(x)))
## Not run:
## below a number of other options are shown
## but they take too long to run

# Change seriation method ############
#seriation::show_seriation_methods("dist")
czek_matrix(x,order = "GW")
czek_matrix(x,order = "ga")
czek_matrix(x,order = sample(1:nrow(x)))

# Change number of classes ############
czek_matrix(x,n_classes = 3)

# Change the partition boundaries ############

#10%, 40% and 50%
czek_matrix(x,interval_breaks = c(0.1,0.4,0.5))

#[0,1] (1,4] (4,6] (6,8.48]
czek_matrix(x,interval_breaks = c(0,1,4,6,8.48))

#[0,1.7] (1.7,3.39]  (3.39,5.09] (5.09,6.78] (6.78,8.48]
czek_matrix(x,interval_breaks = "equal_width_between_classes")

# Change number of classes ############
czek_matrix(x,monitor = TRUE)
```

```
czek_matrix(x,monitor = "cumulativ_plot")

# Change distance function ############
czek_matrix(x,distfun = function(x) stats::dist(x,method = "manhattan"))

# Change dont scale the data ############
czek_matrix(x,scale_data = FALSE)
czek_matrix(stats::dist(x))

# Change additional settings to the seriation method ############
czek_matrix(x,order="ga",control=list(popSize=200, suggestions=c("SPIN_STS","QAP_2SUM")))

# Create matrix as originally described by Czekanowski (1909), with each column
# assigned levels according to how the order statistics of the  distances in it
# are grouped. The grouping below is the one used by Czekanowski (1909).
czek_matrix(x,original_diagram=TRUE,column_order_stat_grouping=c(3,4,5,6))

# Create matrix with two focal object that will not influence seriation
czek_matrix(x,focal_obj=c("Merc 280","Merc 450SL"))
# Same results but with object indices
czek_res<-czek_matrix(x,focal_obj=c(10,13))

# we now place the two objects in a new place
czek_res_neworder<-manual_reorder(czek_res,c(1:10,31,11:20,32,21:30), orig_data=x)

# the same can be alternatively done by hand
attr(czek_res,"order")<-attr(czek_res,"order")[c(1:10,31,11:20,32,21:30)]
# and then correct the values of the different criteria so that they
# are consistent with the new ordering
attr(czek_res,"Path_length")<-seriation::criterion(stats::dist(scale(x)),
order=seriation::ser_permutation(attr(czek_res, "order")),
method="Path_length")

# Here we need to know what criterion was used for the seriation procedure
# If the seriation package was used, then see the manual for seriation::seriate()
# seriation::criterion().
# If the genetic algorithm shipped with RMaCzek was used, then it was the Um factor.
attr(czek_res,"criterion_value")<-seriation::criterion(stats::dist(scale(x)),
order=seriation::ser_permutation(attr(czek_res, "order")),method="Path_length")
attr(czek_res,"Um")<-RMaCzek::Um_factor(stats::dist(scale(x)),
order= attr(czek_res, "order"), inverse_um=FALSE)
# Czekanowski's Clusterings ############
# Exact Clustering
czek_exact = czek_matrix(x, order = "GW", cluster = TRUE, num_cluster = 2, min.size = 2)
plot(czek_exact)
attr(czek_exact, "cluster_type") # To get the clustering type.
attr(czek_exact, "cluster_res") # To get the clustering suggestion.
attr(czek_exact, "membership") # To get the membership matrix

# Fuzzy Clustering
   ## The below example will only work if you have FuzzyDBScan installed
## (from GitHub or CRAN's archive). Otherwise the code stops.
## czek_fuzzy = czek_matrix(x, order = "OLO", cluster = TRUE, num_cluster = 2,
```

```
## cluster_type = "fuzzy", min.size = 2, scale_bandwidth = 0.2)
## plot(czek_fuzzy)
## attr(czek_fuzzy, "cluster_type") # To get the clustering type.
## attr(czek_fuzzy, "cluster_res") # To get the clustering suggestion.
## attr(czek_fuzzy, "membership") # To get the membership matrix

## End(Not run)
```

`internet_availability` *Data of internet_availability*

Description

Data of internet_availability

Usage

```
internet_availability
```

Format

An object of class `list` of length 3.

`manual_reorder` *Manually reorder Czekanowski's Diagram*

Description

This is a function that allows the user to manully reorder Czekanowski's Diagram and recalculates all the factors.

Usage

```
manual_reorder(x, v_neworder, ...)
```

Arguments

`x`	a matrix with class czek_matrix, czek_matrix_dist or data matrix/data.frame or dist object.
`v_neworder`	a numeric vector with the new ordering.
`...`	specifies further parameters that will be passed to the czek_matrix function.

Value

The function returns a Czekanowski's Diagram with the new order and recalculated factors.

Examples

```
# Set data ####
x<-mtcars

# Calculate Czekanowski's diagram
czkm<-czek_matrix(x)
czkm_dist<-czek_matrix(x,as_dist=TRUE)
# new ordering
neworder<-attr(czkm,"order")
neworder[1:2]<-neworder[2:1]
# reorder the diagram
#if the output was Czekanowski's diagram without the distances
#remembered, then the original data has to be passed so that
#factors can be recalculated.
new_czkm<-manual_reorder(czkm,v_neworder=neworder,orig_data=x)
new_czkm_dist<-manual_reorder(czkm_dist,v_neworder=neworder)
#we can also pass the original data directly
new_czkm<-manual_reorder(x,v_neworder=neworder)
#and this is equivalent to calling
czkm<-czek_matrix(x,order=neworder)
#up to the value of the "criterion_value" attribute
#which in the second case can be lost, as no information is passed
#on which one was originally used, while in the first case it might
#be impossible to recalculate-only criteria values from seriate are supported
#if a user has a custom seriation function, then they need to recalculate this
#value themselves
```

`plot.czek_matrix` *Produce a Czekanowski's Diagram*

Description

This is a function that can produce a Czekanowski's Diagram and present clustering findings.

Usage

```
## S3 method for class 'czek_matrix'
plot(
  x,
  values = NULL,
  type = "symbols",
  plot_title = "Czekanowski's diagram",
  tl.cex = 1,
  tl.offset = 0.4,
  tl.srt = 90,
  pal = brewer.pal(n = 8, name = "Dark2"),
  alpha = 0.3,
  ps_power = 0.6,
```

```
  col_size = 1,
  cex.main = 1,
  ...
)
```

Arguments

`x`	a matrix with class czek_matrix.
`values`	specifies the color or the size of the symbols in the graph. The standard setting is a grey scale for a color graph and a vector with the values 2,1,0.5,0.25 and 0 for a graph with symbols.
`type`	specifies if the graph should use color or symbols. The standard setting is symbols.
`plot_title`	specifies the main title in the graph.
`tl.cex`	Numeric, for the size of text label.
`tl.offset`	Numeric, for text label.
`tl.srt`	Numeric, for text label, string rotation in degrees.
`pal`	The colour vector representing the clusters.
`alpha`	Factor modifying the opacity, alpha, typically in [0,1].
`ps_power`	A power value to adjust point size.
`col_size`	When type="col", the size of each point (maximum is 1).
`cex.main`	Specify the size of the title text.
`...`	specifies further parameters that can be passed on to the plot function.

Examples

```
# Set data ####
# Not Cluster
czek = czek_matrix(mtcars)
# Exact Clustering
czek_exact = czek_matrix(mtcars, order = "GW", cluster = TRUE, num_cluster = 2, min.size = 2)

# Standard plot ############
plot(czek_exact)

# Edit diagram title
plot(czek, plot_title = "mtcars", cex.main = 2)

# Change point size ############
# Specify values
plot(czek, values = c(1, 0.8, 0.5, 0.2, 0))
plot(czek, values = grDevices::colorRampPalette(c("black", "red", "white"))(5))

# set point size for 'symbols' type by setting power value
plot(czek, type = "symbols", ps_power = 1)

# set point size for 'col' type
```

```
plot(czek, type = "col", col_size = 0.6)

# Specify type ############
plot(czek, type = "symbols")
plot(czek, type = "col")

# Edit cluster ############
# Edit colors
plot(czek_exact, pal = c("red", "blue"))
# Edit opacity
plot(czek_exact, alpha = 0.5)
# Fuzzy Clustering
## The below example will only work if you have FuzzyDBScan installed
## (from GitHub or CRAN's archive). Otherwise the code stops.
## czek_fuzzy = czek_matrix(mtcars, order = "OLO", cluster = TRUE, num_cluster = 2,
## cluster_type = "fuzzy", min.size = 2, scale_bandwidth = 0.2)
# Standard plot ############
## plot.czek_matrix(czek_fuzzy)
```

`print.czek_matrix` *Prints information concerning Czekanowski's Diagram*

Description

This is a function that prints out information on a Czekanowski's Diagram.

Usage

```
## S3 method for class 'czek_matrix'
print(x, print_raw = FALSE, ...)
```

Arguments

`x`	a matrix with class czek_matrix.
`print_raw`	logical, if TRUE print out raw, as if base::print() was called, in particular this prints out the matrix itself, if FALSE (default) print out a summary. Furthermore, with print_raw=TRUE the attributes "levels", "partition_boundaries" and "n_classes" defining the diagram will be printed out.
`...`	specifies further parameters that can be passed on to the print function.

Value

The function returns a Czekanowski's Diagram.

Examples

```
# Set data ####
x<-czek_matrix(mtcars)


# Standard print ############
print(x)
print.czek_matrix(x)
# Print out the raw object ############
print(x,print_raw=TRUE)
print.czek_matrix(x,print_raw=TRUE)
```

`print.czek_matrix_dist` *Prints information concerning Czekanowski's Diagram*

Description

This is a function that prints out information on a Czekanowski's Diagram, but when the actual distances were saved.

Usage

```
## S3 method for class 'czek_matrix_dist'
print(x, print_raw = FALSE, ...)
```

Arguments

`x`	a matrix with class czek_matrix_dist.
`print_raw`	logical, if TRUE print out raw, as if base::print() was called, in particular this prints out the matrix itself, if FALSE (default) print out a summary. Furthermore, with print_raw=TRUE the attributes "levels", "partition_boundaries" and "n_classes" defining the diagram will be printed out.
`...`	specifies further parameters that can be passed on to the print function.

Value

The function returns a Czekanowski's Diagram.

Examples

```
# Set data ####
x<-czek_matrix(mtcars,as_dist=TRUE)


# Standard print ############
print(x)
print.czek_matrix(x)
```

```
# Print out the raw object ############
print(x,print_raw=TRUE)
print.czek_matrix(x,print_raw=TRUE)
```

`read_maczek_file` *function to load data from an mdt file (MaCzek 3.3 - http://www.antropologia.uw.edu.pl/MaCzek/maczek.html)*

Description

This software comes AS IS in the hope that it will be useful WITHOUT ANY WARRANTY, NOT even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. Please understand that there may still be bugs and errors. Use it at your own risk. We take no responsibility for any errors or omissions in this package or for any misfortune that may befall you or others as a result of its use. Please send comments and report bugs to Krzysztof Bartoszek at krzbar@protonmail.ch .

Usage

```
read_maczek_file(filepath)
```

Arguments

`filepath`	path to file *.mdt.

Value

data.frame.

Author(s)

Piotr Jaskulski

`RMaCzek` *RMaCzek: A package that can produce Czekanowski's diagram*

Description

This package produces Czekanowski's diagram

The packages functions

czek_matrix A function that returns a distance matrix where the distances are divided into classes. It also offers exact and fuzzy Czekanowski's clustering algorithm. The return from the function is expected to be passed into the plot function.

plot.czek_matrix A function that returns Czekanowski's Diagram.

Author(s)

Albin Vasterlund, Krzysztof Bartoszek, Ying Luo

`seals_similarities` *Data of seals_similarities*

Description

Data of seals_similarities

Usage

```
seals_similarities
```

Format

An object of class `matrix` (inherits from `array`) with 37 rows and 37 columns.

`skulls_distances` *Data of skulls_distances*

Description

Data of skulls_distances

Usage

```
skulls_distances
```

Format

An object of class `matrix` (inherits from `array`) with 13 rows and 13 columns.

`Um_factor` *Calculate the Um factor*

Description

The function calculates the Um factor associated with an ordering of the rows and columns of a distance matrix. Lower values indicate a better grouping of similar objects. This was the original objective function proposed in the MaCzek program for producing Czekanowski's Diagram.

Usage

```
Um_factor(
  distMatrix,
  order = NULL,
  matrix_conversion_coefficient = 1,
  inverse_um = TRUE
)
```

Arguments

`distMatrix`	a 'dist' object, matrix of distances between observations.
`order`	a vector, if NULL, then the value of the factor is calculate for the distance matrix as is, otherwise the rows and columns are reordered according to the vector order.
`matrix_conversion_coefficient`	numeric, value to be added to the distances, so that a division by 0 error is not thrown.
`inverse_um`	logical, if TRUE, then the negative is returned. Default TRUE as the function is called in the genetic algorithm maximization procedures.

Value

The function returns a numeric value equalling the Um_factor.

Examples

```
# Set data ####
x<-mtcars

mD<-stats::dist(scale(x))
mCz<-czek_matrix(x)
Um_factor(mD)
Um_factor(mD,order=attr(mCz,"order"))
```

`urns` *Data of urns*

Description

Data of urns

Usage

```
urns
```

Format

An object of class `matrix` (inherits from `array`) with 15 rows and 9 columns.

Index