Package 'archetypal'

June 13, 2019

Version 1.0.0

Title Finds the Archetypal Analysis of a Data Frame

Description Performs archetypal analysis by using Convex Hull approximation under a full control of all algorithmic parameters.
It contains functions useful for finding the proper initial approximation, the optimal number of archetypes and function for applying main algorithm.
Morup, M., Hansen, LK (2012) <doi:10.1016/j.neucom.2011.06.033>.
Hochbaum, DS, Shmoys, DB (1985) <doi:10.1287/moor.10.2.180>.
Eddy, WF(1977) <doi:10.1145/355759.355768>.
Barber, CB, Dobkin, DP, Huhdanpaa, HT (1996) <doi:10.1145/235815.235821>.
Christopoulos, DT (2016) <doi:10.2139/ssrn.3043076> .

Maintainer Demetris Christopoulos <dchristop@econ.uoa.gr>

Depends R (>= 3.1.0)

Imports Matrix, geometry, inflection, doParallel, lpSolve, methods

Suggests knitr, rmarkdown, plot3D

VignetteBuilder knitr

License GPL (>= 2)

Encoding UTF-8

LazyData true

ByteCompile true

NeedsCompilation no

Author Demetris Christopoulos [aut, cre],
David Midgley [ctb],
INSEAD Fontainebleau France [fnd, cph]

Repository CRAN

Date/Publication 2019-06-13 13:20:07 UTC

R topics documented:

`archetypal-package` *Finds the Archetypal Analysis of a Data Frame*

Description

Performs archetypal analysis by using Convex Hull approximation under a full control of all algorithmic parameters. It contains functions useful for finding the proper initial approximation, the optimal number of archetypes and function for applying main algorithm.

Compute Archerypal Analysis (AA)

The main function is `archetypal` which is a modification of PCHA algorithm, see [1], [2], suitable for R language. It provides control to the entrire set of involved parameters and has two main options:

1. initialrows=NULL, then a method from 'projected_convexhull', 'convexhull', 'partitioned_convexhull', 'furthestsum', 'outmost', 'random' is used
2. initialrows=a vector of kappas rows, then given rows form the initial solution for AA

This is the main function of the package, but extensive trials has shown that:

- AA may be very difficult to run if a random initial solution has beeen chosen
- for the same data set the final Sum of Squared Errors (SSE) may be much smaller if initial solution is close to the final one
- even the quality of AA done is affected from the starting point

This is the reason why we have developed a whole set of methods for choosing initial solutionfor main PCHA algorithm.

Find a time efficient initial approximation for AA

There are three functions that work with the Convex Hull (CH) of data set.

1. `find_outmost_convexhull_points` computes the CH of all points
2. `find_outmost_projected_convexhull_points` computes the CH for all possible combinations of variables taken by n (default=2)
3. `find_outmost_partitioned_convexhull_points` makes partitions of data frame, then computes CH for each partition and finally gives the CH of overall union

The most simple method for estimating an initial solution is `find_outmost_points` where we just compute the "outmost points", ie those that are the most frequent outmost for all available points.

The default method "FurthestSum" (FS) of PCHA (see [1], [2]) is used by `find_furthestsum_points` which applies FS for many times (default=10) and then finds the most frequent points.

Of course "random" method is available for comparison reasons and that gives a random set of kappas points as initial solution.

All methods give the number of rows for the input data frame.

Find the optimal number of archetypes (kappas)

For that task `find_optimal_kappas` is available which runs for each kappas from 1 to maxKappas (default=15) ntrials (desault=10) times AA, stores SSE, VarianceExplained from each run and then computes knee point by nusing UIK method, see [3].

Evaluate the quality of Archerypal Analysis (AA)

By using function `check_Bmatrix` we can evaluate the overall quality of applied method and algorithm. Quality can be considered high:

1. if every archetype is being created by a small number of points
2. if relevant weights are not numerically insignificant

Of course we must take into account the SSE and VarianceExplained, but if we have to compare two solutions with similar termination status, then we must choose that of the simplest B matrix form.

Note

Bug reports and feature requests can be sent to <dchristop@econ.uoa.gr>.

Author(s)

Maintainer: Demetris Christopoulos <dchristop@econ.uoa.gr>

Other contributors:

- David Midgley <david.midgley@insead.edu> [contributor]
- INSEAD Fontainebleau France [funder, copyright holder]

References

[1] M Morup and LK Hansen, "Archetypal analysis for machine learning and data mining", Neurocomputing (Elsevier, 2012). https://doi.org/10.1016/j.neucom.2011.06.033.

[2] Source: http://www.mortenmorup.dk/index_files/Page327.htm , last accessed 2019-06-07

[3] Christopoulos, Demetris T., Introducing Unit Invariant Knee (UIK) As an Objective Choice for Elbow Point in Multivariate Data Analysis Techniques (March 1, 2016). Available at SSRN: https://ssrn.com/abstract=3043076 or http://dx.doi.org/10.2139/ssrn.3043076

See Also

archetypal

`align_archetypes_from_list` *Align archetypes from a list either by the most frequent found or by using a given archetype*

Description

Align archetypes from a list either by the most frequent or by using a given archetype

Usage

```
align_archetypes_from_list(archs_list, given_arch = NULL,
  varnames = NULL, ndigits = 0, parallel = FALSE, nworkers = NULL)
```

Arguments

`archs_list`	The list of archetypes that must be aligned
`given_arch`	If it is not NULL, then the given archetype will by used as guide for aligning other archetypes of list. Otherwise, a heuristic for finding the most frequent archetype will be used.
`varnames`	The character vector of variable names that must be used. If it is NULL, then the column names of first archetype will be used.
`ndigits`	The character of digits that will be used for truncation.
`parallel`	If it set to TRUE, then parallel processing will be applied.
`nworkers`	The number of logical processors that will be used for parallel computing (usually it is the double of available physical cores)

Value

A list with members:

1. arch_guide, the archetype used as guide for aligning others
2. phrases_most, a table with all rounded phrases from archetypes. Frequencies are in decreasing order,so first row indicates the most frequent sequence, if exists. Otherwise we take randomly a case and proceed.
3. archs_aa_output, a data frame with rows all given archetypes
4. archs_aligned, the final list of aligned archetypes

References

This function is a modifcation of "align_arc" function from package "ParetoTI", see https://github.com/vitkl/ParetoTI and https://github.com/vitkl/ParetoTI/blob/master/R/align_arc.R

Examples

```
data("wd2") #2D demo
df=wd2
# Define 4 archetypes found for it
dalist=list(c(2.172991,3.200754,5.384013,2.579770,4.860343,3.085111),
            c(5.430821,3.128493,2.043495,3.146342,4.781851,2.710885),
            c(5.430752,2.043403,3.128520,3.146252,2.710979,4.781880),
            c(2.043854,5.430890,3.127183,2.710522,3.146432,4.780432))
archslist=lapply(dalist, function(x){matrix(x,ncol=2)}) #not aligned
# Run aligner
yy=align_archetypes_from_list(archs_list = archslist,given_arch = archslist[[1]])
yy$arch_guide
aligned_archs=yy$archs_aligned
aligned_archs #observe thatn they are comparabnle now
```

`archetypal` *archetypal: Finds the archetypal analysis of a data frame by using a modified version of PCHA algorithm*

Description

Performs archetypal analysis by using Convex Hull approximation under a full control of all algorithmic parameters.

Usage

```
archetypal(df, kappas, initialrows = NULL,
  method = "projected_convexhull", nprojected = 2, npartition = 10,
  nfurthest = 10, maxiter = 2000, conv_crit = 1e-06,
  var_crit = 0.9999, verbose = TRUE, rseed = NULL, aupdate1 = 25,
```

```
aupdate2 = 10, bupdate = 10, muAup = 1.2, muAdown = 0.5,
muBup = 1.2, muBdown = 0.5, SSE_A_conv = 1e-09,
SSE_B_conv = 1e-09, save_history = FALSE, nworkers = NULL)
```

Arguments

df	The data frame with dimensions n x d
kappas	The number of archetypes
initialrows	The initial set of rows from data frame that will be used for starting algorithm
method	The method that will be used for computing initial approximation: 1. projected_convexhull, see find_outmost_projected_convexhull_points 2. convexhull, see find_outmost_convexhull_points 3. partitioned_convexhull, see find_outmost_partitioned_convexhull_points 4. furthestsum, see find_furthestsum_points 5. outmost, see find_outmost_points 6. random, a random set of kappas points will be used
nprojected	The dimension of the projected subspace for find_outmost_projected_convexhull_points
npartition	The number of partitions for find_outmost_partitioned_convexhull_points
nfurthest	The number of times that FurthestSum algorithm will be applied
maxiter	The maximum number of iterations for main algorithm application
conv_crit	The SSE convergence criterion of termination: iterate until |dSSE|/SSE<conv_crit
var_crit	The Variance Explained (VarExpl) convergence criterion of termination: iterate until VarExpl<var_crit
verbose	If it is set to TRUE, then both initialization and iteration details are printed out
rseed	The random seed that will be used for seting initial A matrix. Useful for reproducible results.
aupdate1	The number of intial applications of Aupdate for improving the initially randomly selected A matrix
aupdate2	The number of Aupdate applications in main iteration
bupdate	The number of Bupdate applications in main iteration
muAup	The factor (>1) by which muA is multiplied when it holds SSE<=SSE_old(1+SSE_A_conv)
muAdown	The factor (<1) by which muA is multiplied when it holds SSE>SSE_old(1+SSE_A_conv)
muBup	The factor (>1) by which muB is multiplied when it holds SSE<=SSE_old(1+SSE_B_conv)
muBdown	The factor (<1) by which muB is multiplied when it holds SSE>SSE_old(1+SSE_B_conv)
SSE_A_conv	The convergence value used in SSE<=SSE_old(1+SSE_A_conv). Warning: there exists a Matlab crash sometimes after setting this to 1E-16 or lower
SSE_B_conv	The convergence value used in SSE<=SSE_old(1+SSE_A_conv). Warning: there exists a Matlab crash sometimes after setting this to 1E-16 or lower
save_history	If set TRUE, then iteration history is being saved for further use
nworkers	The number of logical processors that will be used for parallel computing (usually it is the double of available physical cores)

Value

A list with members:

1. BY, the $kappas \times d$ matrix of archetypes found
2. A, the $n \times kappas$ matrix such that Y ~ Y - ABY or Frobenius norm ||Y-ABY|| is minimum
3. B, the $kappas \times n$ matrix such that Y ~ Y - ABY or Frobenius norm ||Y-ABY|| is minimum
4. SSE, the sum of squared error SSE = ||Y-ABY||^2
5. varexpl, the Variance Explained =(SST-SSE)/SST where SST is the total sum of squares for data set matrix
6. initialsolution, the initially used set of rows from data frame in order to start the algorithm
7. freqstable, the frequency table for all found rows, if it is available.
8. iterations, the number of main iterations done by algorithm
9. time, the time in seconds that was spent from entire run
10. converges, if it is TRUE, then convegence was achieved before the end of maximum allowed iterations
11. nAup, the total number of times when it was SSE<=SSE_old(1+SSE_A_conv) in Aupdate processes. Useful for debugging purposes.
12. nAdown The total number of times when it was SSE>SSE_old(1+SSE_A_conv) in Aupdate processes. Useful for debugging purposes.
13. nBup, the total number of times when it was SSE<=SSE_old(1+SSE_B_conv) in Bupdate processes. Useful for debugging purposes.
14. nBdown, the total number of times when it was SSE>SSE_old(1+SSE_A_conv in Bupdate processes. Useful for debugging purposes.
15. run_results, a list of iteration related details: SSE, varexpl, time, B, BY for all iterations done.

References

[1] M Morup and LK Hansen, "Archetypal analysis for machine learning and data mining", Neurocomputing (Elsevier, 2012). https://doi.org/10.1016/j.neucom.2011.06.033.

[2] Source: http://www.mortenmorup.dk/index_files/Page327.htm , last accessed 2019-06-07

Examples

```
# Create a small 2D data set from 3 corner-points:
p1=c(1,2);p2=c(3,5);p3=c(7,3)
dp=rbind(p1,p2,p3);dp
set.seed(916070)
pts=t(sapply(1:20, function(i,dp){
  cc=runif(3)
  cc=cc/sum(cc)
  colSums(dp*cc)
},dp))
df=data.frame(pts)
colnames(df)=c("x","y")
# Run AA:
```

```
aa=archetypal(df=df,kappas = 3,verbose=FALSE,save_history = TRUE)
# Archetypes:
archs=data.frame(aa$BY)
archs
# See main results:
names(aa)
aa[c("SSE","varexpl","iterations","time")]
# See history of iterations:
names(aa$run_results)
```

`check_Bmatrix` *Function which checks B matrix of Archetypal Analysis Y ~ A B Y in order to find the used rows for creating each archetype and the relevant used weights.*

Description

Function which checks B matrix of Archetypal Analysis Y ~ A B Y in order to find the used rows for creating each archetype and the relevant used weights.

Usage

```
check_Bmatrix(B, print.details = TRUE)
```

Arguments

`B`	The $kappas \times n$ matrix such that Y ~ Y - ABY or Frobenius norm ‖Y-ABY‖ is minimum
`print.details`	If set to TRUE, then results are printed out.

Value

A list with members:

1. used_rows, a list with used rows for creating each archetype
2. used_weights, a list with the relevant weights that have been used

See Also

`archetypal`

Examples

```
# Create a small 2D data set from 3 corner-points:
p1=c(1,2);p2=c(3,5);p3=c(7,3)
dp=rbind(p1,p2,p3);dp
set.seed(916070)
pts=t(sapply(1:20, function(i,dp){
  cc=runif(3)
```

```
  cc=cc/sum(cc)
  colSums(dp*cc)
},dp))
df=data.frame(pts)
colnames(df)=c("x","y")
# Run AA:
aa=archetypal(df=df,kappas = 3,verbose=FALSE)
# Check B matrix:
B=aa$B
yy=check_Bmatrix(B)
yy$used_rows
yy$used_weights
#
```

`find_furthestsum_points` *Function which finds the furthest sum points in order to be used as initial solution in archetypal analysis*

Description

Function which finds the furthest sum points in order to be used as initial solution in archetypal analysis

Usage

```
find_furthestsum_points(df, kappas, nfurthest = 100, nworkers = 10,
  sortrows = TRUE)
```

Arguments

`df`	The data frame with dimensions n x d
`kappas`	The number of archetypes
`nfurthest`	The number of applications for FurthesSum algorithm
`nworkers`	The number of logical processors that will be used
`sortrows`	If it is TRUE, then rows will be sorted

Value

A list with members:

1. outmost, the first kappas furthest sum points as rows of data frame
2. outmostall, all the furthest sum points that have been found as rows of data frame
3. outmostfrequency, a matrix with frequency and cumulative frequency for furthest sum rows

See Also

`FurthestSum`

Examples

```
data("wd3") #3D demo
df=wd3
yy=find_furthestsum_points(df,kappas = 4,nfurthest = 10,nworkers = 1)
yy$outmost
yy$outmostall
yy$outmostfrequency
```

`find_optimal_kappas` *Function for finding the optimal number of archetypes*

Description

Function for finding the optimal number of archetypes in order to apply Archetypal Analysis for a data frame.

Usage

```
find_optimal_kappas(df, maxkappas = 15,
  method = "projected_convexhull", ntrials = 10, nworkers = NULL)
```

Arguments

`df`	The data frame with dimensions $n \times d$
`maxkappas`	The maximum number of archetypes for which algorithm will be applied
`method`	The method that will be used for computing the initial solution
`ntrials`	The number of times that algorithm will be applied for each kappas
`nworkers`	The number of logical processors that will be used for parallel computing (usually it is the double of available physical cores)

Details

After having found the SSE for each kappas, method UIK (see [1]) is used for estimating the knee point as the optimal kappas.

Value

A list with members:

1. all_sse, all available SSE for all kappas and all trials per kappas
2. all_sse1, all available SSE(k)/SSE(1) for all kappas and all trials per kappas
3. bestfit_sse, only the best fit SSE trial for each kappas
4. bestfit_sse1, only the best fit SSE(k)/SSE(1) trial for each kappas
5. all_kappas, the knee point of scree plot for all 4 SSE results
6. optimal_kappas, the knee point from best fit SSE results

References

[1] Christopoulos, Demetris T., Introducing Unit Invariant Knee (UIK) As an Objective Choice for Elbow Point in Multivariate Data Analysis Techniques (March 1, 2016). Available at SSRN: https://ssrn.com/abstract=3043076 or http://dx.doi.org/10.2139/ssrn.3043076

See Also

archetypal

Examples

```
{
# Run may take a while...
# Create data frame:
t1=Sys.time()
p1=c(1,2);p2=c(3,5);p3=c(7,3)
dp=rbind(p1,p2,p3);dp
set.seed(916070)
pts=t(sapply(1:20, function(i,dp){
  cc=runif(3)
  cc=cc/sum(cc)
  colSums(dp*cc)
},dp))
df=data.frame(pts)
colnames(df)=c("x","y")
# Run:
yy=find_optimal_kappas(df,maxkappas = 10)
# Results:
names(yy)
# Best fit SSE:
yy$bestfit_sse
# Optimal kappas from UIK method:
yy$optimal_kappas
#

}
```

---

find_outmost_convexhull_points

Function which finds the outmost convex hull points in order to be used as initial solution in archetypal analysis

---

Description

Function which finds the outmost convex hull points in order to be used as initial solution in archetypal analysis

Usage

```
find_outmost_convexhull_points(df, kappas)
```

Arguments

df	The data frame with dimensions n x d
kappas	The number of archetypes

Details

This function uses the chull when n=2 (see [1], [2]) and the convhulln for n>2 (see [3]) cases.

Value

A list with members:

1. outmost, the first kappas outmost points as rows of data frame
2. outmostall, all the outmost points that have been found as rows of data frame
3. outmostfrequency, a matrix with frequency and cumulative frequency for outmost rows

References

[1] Eddy, W. F. (1977). A new convex hull algorithm for planar sets. ACM Transactions on Mathematical Software, 3, 398-403. doi: 10.1145/355759.355766.

[2] Eddy, W. F. (1977). Algorithm 523: CONVEX, A new convex hull algorithm for planar sets [Z]. ACM Transactions on Mathematical Software, 3, 411-412. doi: 10.1145/355759.355768.

[3] Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T., "The Quickhull algorithm for convex hulls" ACM Trans. on Mathematical Software, 22(4):469-483, Dec 1996, http://www.qhull.org

See Also

find_furthestsum_points, find_outmost_projected_convexhull_points, find_outmost_partitioned_convexhull_points & find_outmost_points

Examples

```
data("wd2") #2D demo
df=wd2
yy=find_outmost_convexhull_points(df,kappas=3)
yy$outmost #the rows of 3 outmost points
df[yy$outmost,] #the 3 outmost points
yy$outmostall #all outmost cH rows
yy$outmostfrequency #their frequency
#
###
#
data("wd3") #3D demo
df=wd3
yy=find_outmost_convexhull_points(df,kappas=4)
yy$outmost #the rows of 4 outmost points
df[yy$outmost,] #the 4 outmost points
yy$outmostall #all outmost cH rows
yy$outmostfrequency #their frequency
```

`find_outmost_partitioned_convexhull_points` *Function which finds the outmost convex hull points after making np samples and findix convex hull for each of them. To be used as initial solution in archetypal analysis.*

Description

Function which finds the outmost convex hull points after making np samples and findix convex hull for each of them. To be used as initial solution in archetypal analysis

Usage

```
find_outmost_partitioned_convexhull_points(df, kappas, np = 10,
  nworkers = NULL)
```

Arguments

`df`	The data frame with dimensions n x d
`kappas`	The number of archetypes
`np`	The number of partitions that will be used (or the number of samples)
`nworkers`	The number of logical processors that will be used

Value

A list with members:

1. outmost, the first kappas outmost points as rows of data frame
2. outmostall, all the outmost points that have been found as rows of data frame
3. outmostfrequency, a matrix with frequency and cumulative frequency for outmost rows

See Also

`find_furthestsum_points`, `find_outmost_projected_convexhull_points`, `find_outmost_convexhull_points` & `find_outmost_points`

Examples

```
data("wd2") #2D demo
df=wd2
yy=find_outmost_partitioned_convexhull_points(df,kappas=3,nworkers = 1)
yy$outmost #the rows of 3 outmost points
df[yy$outmost,] #the 3 outmost points
yy$outmostall #all outmost rows
yy$outmostfrequency #their frequency
```

`find_outmost_points` *Function which finds the outmost points in order to be used as initial solution in archetypal analysis*

Description

Function which finds the outmost points in order to be used as initial solution in archetypal analysis

Usage

```
find_outmost_points(df, kappas)
```

Arguments

`df`	The data frame with dimensions n x d
`kappas`	The number of archetypes

Value

A list with members:

1. outmost, the first kappas outmost points as rows of data frame
2. outmostall, all the outmost points that have been found as rows of data frame
3. outmostfrequency, a matrix with frequency and cumulative frequency for outmost rows

Warning

This is a rather naive way to find the outmost points of a data frame and it should be used with caution since for a n x d matrix we need in general 8 n^2/(2^30) Gb of memory RAM for numeric case. Check your machine and use it.

See Also

`find_furthestsum_points`, `find_outmost_convexhull_points`,
`find_outmost_projected_convexhull_points` & `find_outmost_partitioned_convexhull_points`

Examples

```
data("wd2") #2D demo
df=wd2
yy=find_outmost_points(df,kappas=3)
yy$outmost #the rows of 3 outmost points
yy$outmostall #all outmost found
yy$outmostfrequency #frequency table for all
df[yy$outmost,] #the 3 outmost points
#
###
#
```

```
data("wd3") #3D demo
df=wd3
yy=find_outmost_points(df,kappas=4)
yy$outmost #the rows of 4 outmost points
yy$outmostall #all outmost found
yy$outmostfrequency #frequency table for all
df[yy$outmost,] #the 4 outmost points
```

`find_outmost_projected_convexhull_points` *Function which finds the outmost projected convex hull points in order to be used as initial solution in archetypal analysis*

Description

Function which finds the outmost projected convex hull points in order to be used as initial solution in archetypal analysis

Usage

```
find_outmost_projected_convexhull_points(df, kappas, n = 2)
```

Arguments

`df`	The data frame with dimensions n x d
`kappas`	The dimension of the projected subspace
`n`	The number of archetypes

Details

This function uses the `chull` when n=2 and the `convhulln` for n>2 cases. See [1] and [2] respectively for more details.

Value

A list with members:

1. outmost, the first kappas outmost points as rows of data frame
2. outmostall, all the outmost points that have been found as rows of data frame
3. outmostfrequency, a matrix with frequency and cumulative frequency for outmost rows

References

[1] Eddy, W. F. (1977). Algorithm 523: CONVEX, A new convex hull algorithm for planar sets. ACM Transactions on Mathematical Software, 3, 411-412. doi: 10.1145/355759.355768.

[2] Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T., "The Quickhull algorithm for convex hulls" ACM Trans. on Mathematical Software, 22(4):469-483, Dec 1996, http://www.qhull.org

See Also

find_furthestsum_points, find_outmost_convexhull_points

find_outmost_partitioned_convexhull_points & find_outmost_points

Examples

```
data("wd2") #2D demo
df=wd2
yy=find_outmost_projected_convexhull_points(df,kappas=3)
yy$outmost #the rows of 3 outmost projected convexhull points
yy$outmostall #all outmost found
yy$outmostfrequency #frequency table for all
df[yy$outmost,] #the 3 outmost projected convexhull points
#
###
#
data("wd3") #3D demo
df=wd3
yy=find_outmost_projected_convexhull_points(df,kappas=4)
yy$outmost #the rows of 4 outmost projected convexhull points
yy$outmostall #all outmost found
yy$outmostfrequency #frequency table for all
df[yy$outmost,] #the 4 outmost projected convexhull points
```

`FurthestSum` *Application of FurthestSum Algorithm in Order to Find an Intial Solution for Archetypal Analyis*

Description

The FurthestSum algorithm as was written by Morup and Hansen in Matlab, see [1] and it is based on [2]. The algorithm has been converted in order to use commonly used data frames in R.

Usage

```
FurthestSum(Y, kappas, irows, exclude = NULL)
```

Arguments

`Y`	The data frame with dimensions $n \times d$
`kappas`	The number of archetypes
`irows`	The initially used rows of data frame for starting algorithm
`exclude`	The rows of data frame that we want to exclude from being checked

Value

The vector of rows that constitute the initial FurthestSum solution

References

[1] Source: http://www.mortenmorup.dk/index_files/Page327.htm , last accessed 2019-06-07

[2] D.S.Hochbaum,D.B.Shmoys,A best possible heuristic for the k-center problem,

Math.Oper.Res.10(2)(1985)180–184. https://doi.org/10.1287/moor.10.2.180

See Also

find_furthestsum_points.

Examples

```
data("wd3") #3D demo
df=wd3
FurthestSum(df,kappas=4,irows=sample(1:dim(df)[1],1))
```

---

wd2 *2D data set for demonstration purposes*

Description

A data frame of 100 2D points

Usage

```
data("wd2")
```

Format

matrix 100 x 2

Examples

```
data("wd2")
summary(wd2)
```

wd3 *3D data set for demonstration purposes*

Description

A data frame of 100 3D points

Usage

```
data("wd3")
```

Format

matrix 100 x 3

Examples

```
data("wd3")
summary(wd3)
```

Index