Package 'GeomArchetypal'

July 21, 2025

Type Package

Title Finds the Geometrical Archetypal Analysis of a Data Frame

Version 1.0.3

Date 2024-10-20

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

Description Performs Geometrical Archetypal Analysis after creating Grid Archetypes
which are the Cartesian Product of all minimum, maximum variable values. Since the archetypes
are fixed now, we have the ability to compute the convex composition coefficients for all
our available data points much faster by using the half part of Principal Convex Hull Archetypal method.
Additionally we can decide to keep as archetypes the closer to the Grid Archetypes ones. Finally
the number of archetypes is always 2 to the power of the dimension of our data points
if we consider them as a vector space.
Cutler, A., Breiman, L. (1994) <doi:10.1080/00401706.1994.10485840>.
Morup, M., Hansen, LK. (2012) <doi:10.1016/j.neucom.2011.06.033>.
Christopoulos, DT. (2024) <doi:10.13140/RG.2.2.14030.88642>.

License GPL (>= 2)

Depends R (>= 3.1.0)

Imports Matrix, geometry, archetypal, doParallel, methods, plot3D,
distances, rlang, magrittr, dplyr, mirai, abind, scales

Suggests knitr, rmarkdown

VignetteBuilder knitr

Author Demetris Christopoulos [aut, cre, cph],
David Midgley [ctb, cph],
Sunil Venaik [ctb],
INSEAD Hoffmann Institute France [fnd],
The University of Queensland Australia [fnd]

NeedsCompilation no

Language en-US

Repository CRAN

Date/Publication 2024-10-20 10:50:09 UTC

Contents

`GeomArchetypal-package`

Finds the Geometrical Archetypal Analysis of a Data Frame

Description

Performs Geometrical Archetypal Analysis after creating Grid Archetypes which are the Cartesian Product of all minimum, maximum variable values. Since the archetypes are fixed now, we have the ability to compute the convex composition coefficients for all our available data points much faster by using the half part of PCHA method. Additionally we can decide to keep as archetypes the closer to the Grid Archetypes ones. Finally the number of archetypes is always 2 to the power of the dimension of our data points if we consider them as a vector space.

Details

Given a data frame `df` which is a matrix of n observations (rows) for the d variables (columns) we compute for all variables Xj the (Xj.min , Xj.max), j=1,2,..., n.

By taking the Cartesian Product of all those sets we form the vector set of Grid Archetypes which are 2 to the power of d and all other points lie inside their Convex Hull.

For example if we take the case of d=2 and our variables are named X,Y, then the Cartesian Product gives next points:

(Xmin,Ymin),(Xmax,Ymin),(Xmin,Ymax),(Xmax,Ymax)

Now the problem of seeking for the best number of archetypes is solved and kappas is 2 to the power of d.

The main task is to express all inner data points as convex combination of the Grid Archetypes. For that reason we drop the half part of the PCHA algorithm of [1], [2] and keep only the desired one, that of the A-matrix computation.
This is the task for the `grid_archetypal()` function.

If we want to seek for the closer to the Grid Archetypes points and set them as archetypes and proceed by the same way, then we use the `closer_grid_archetypal()` function.

All the two above functions use the generic function `fast_archetypal()` which computes the A-matrix for a given set rows of archetypes for our data frame of interest.

Finally we introduce the function `points_inside_convex_hull()` which for a given data frame `df` and a given set of at least `d + 1` archetypes or points in general `dp`, computes the percentage of data points that lie inside the `Convex Hull` which is created by `dp`.
A pdf version of a detailed description can be found in [3].

Author(s)

Demetris Christopoulos
https://orcid.org/0009-0008-6436-095X

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: https://mortenmorup.dk/?page_id=2 , last accessed 2024-03-09
[3] Christopoulos, DT. (2024) https://doi.org/10.13140/RG.2.2.14030.88642

See Also

`grid_archetypal`, `closer_grid_archetypal`, `fast_archetypal`

Examples

```
# Load package
library(GeomArchetypal)
# Create random data
vseed = 20140519
set.seed(vseed)
df=matrix(runif(90) , nrow = 30, ncol=3)
# Grid Archetypal
gaa=grid_archetypal(df, diag_less = 1e-6,
                    niter = 50, use_seed = vseed)
# Print
print(gaa)
# Summary
summary(gaa)
plot(gaa)
# Closer Grid Archetypal
cga=closer_grid_archetypal(df, diag_less = 1e-3,
                           niter = 200, use_seed = vseed)
```

```
# Print
print(cga)
# Summary
summary(cga)
# Plot
plot(cga)
# Fast Archetypal:
# we use as archetyupal rows the closer to the Grid Archetypes
# as they were foind by closer_grid_archetypal() function
fa=fast_archetypal(df, irows = cga$grid_rows, diag_less = 1e-3,
                   niter = 200, use_seed = vseed)
# Print
print(fa)
# Summary
summary(fa)
# Plot
plot(fa)
```

BLB_archetypal *Archetypal Analysis using the Bag of Little Bootstraps*

Description

Archetypal analysis using the bag of little bootstraps as the resampling approach, following [1]

Usage

```
BLB_archetypal(df = NULL, ss_size = NULL, bs_size = NULL,
arches = NULL, use_seed = NULL, n = 20,
r = 100, n_core = 1, n_iter = 100, ci_sigma = 2.5757,
n_tails = 10, diag_less = 0.01)
```

Arguments

df	The data frame with the original sample to be processed.
ss_size	The size of the subsample to be drawn from df without replacement.
bs_size	The size of the bootstrap replications to be drawn with replacement from each subsample, typically nrow(df), but see Details.
arches	A data frame of the archetype profiles, whose column names correspond to variables found in df. See Details.
use_seed	Integer, if not NULL, used as set.seed() for reproducibility.
n	Integer, the number of subsamples to draw from df (default 20).
r	Integer, the number of bootstrap replications of each subsample (default 100). r must be an integer multiple of n_core. See Details.
n_core	Integer, the number of cores available for batch processing of bootstrap replications.

n_iter	Integer, number of iterations for fast_archetypal.
ci_sigma	Numeric, for empirical confidence intervals (default 2.5757 for 99
n_tails	Integer, minimum number of bootstrap estimates required in the tails for robust confidence intervals (default 10 each tail).
diag_less	Test of convergence for the archetypal row weights. The expected mean distance from 1 for the weights of the archetypes themselves (default 0.01). See Details.

Details

Usage. BLB_archetypal is designed to be used with large samples (> 1k cases) and to minimize runtimes by employing parallel processing with multiple cores.
Archetypes. BLB_archetypal is also designed to be used with fixed archetypes which the user has determined a priori. Resampling variability then solely concerns the row weights based on these archetypes, considerably reducing runtimes and simplifying analysis because all bootstraps have the same frame of reference. There are several ways of determining these archetypes, one recommendation that facilitates interpretation is to base them on all combinations of the maximums and minimums of the variables on which the archetypes are to be based. See GeomArchetypal.
Computational process. n subsamples of size ss_size are drawn without replacement from df, r replications of size bs_size are then generated from each subsample with replacement. The fast_archetypal function is then applied to each of the resulting n * r bootstraps to estimate the row weights for the user supplied arches. The mean weights from the n * r bootstraps provide the population estimate for the row weights, and the ci_sigma confidence intervals for this estimate are determined directly from the distribution of bootstrap estimates. The confidence intervals are thus not bias adjusted, but this is less necessary with large samples and the direct method reduces runtimes.
Convergence. In fast_archetypal, the set of archetypes are appended to the data frame to be analyzed. At convergence the row weights for these archetypes should be a target matrix with 1s on the diagonal and 0s off diagonal. diag_less provides the test of this convergence, with the default 0.01 implying the mean of the distances between the set of weights and the target matrix must be <= 0.01. Users may wish to consider stricter tests, such as 0.001 or 1e-6 but these will considerably increase the number of iterations needed before the algorithm converges and hence runtimes. Note if one or more bootstraps do not converge, the function will generate an error as otherwise incorrect estimates would be included in the subsequent calculations. In this case, the user should increase n_iter before rerunning.
Confidence intervals. The default settings of n = 20 subsamples each with r = 100 replications generate 2000 bootstraps and ci_sigma = 2.5757 or ~ 99 Parallel processing. To reduce run times parallel processing can be used, with n_core processing cores. In this case bootstraps are processed in n_core batches and r must be an integral multiple of n_core or an error will occur. From the first batch, the function generates a message with an approximate estimate of the total run time, allowing the user to assess when completion is likely.
Bootstrap size. Typically, this is the size of the original sample. However, if the original sample is very large (> 100k), the user may wish to consider taking a large subsample of the original externally to the function, to avoid excessive runtimes.

Value

An object of class "BLB_archetypal" which is a list with next members:

1. arches, the user supplied archetypes on which the results are based.
2. pop_compos, the population estimates of the compositions (archetypal case weights) for each row of df.
3. lower_ci, the lower confidence interval for these estimates at the ci_sigma level.
4. upper_ci, the upper confidence interval for these estimates at the ci_sigma level.
5. ci_sigma, the ci_sigma level for confidence intervals.
6. N, the original sample size, nrow(df).

Author(s)

David. F. Midgley

References

[1] Ariel Kleiner, Ameet Talwalkar, Purnamrita Sarkar, Michael I. Jordan, doi:10.1111/rssb.12050

See Also

closer_grid_archetypal, grid_archetypal, fast_archetypal

Examples

```
{
library(GeomArchetypal)
library(mirai)
library(parallel)
data("gallupGPS6")
aa_var <- c("patience","risktaking","trust") # variables for archetypal analysis
# define the 2^3 archetypes from minimums and maximums of the data
min_var <- apply(gallupGPS6[, aa_var], 2, min, na.rm = TRUE)
max_var <- apply(gallupGPS6[, aa_var], 2, max, na.rm = TRUE)
temp <- as.data.frame.matrix(rbind(min_var, max_var))
colnames(temp) <- aa_var
list_minmax <- apply(temp, 2, as.list)
rm(temp, min_var, max_var)
arches <- data.matrix(do.call(expand.grid, list_minmax))
arches <- as.data.frame(arches)
rm(list_minmax)
# apply BLB archetypal for a minimal example
test <- BLB_archetypal(df = gallupGPS6,
                       ss_size = 50,
                       bs_size = nrow(gallupGPS6),
                       arches = arches,
                       use_seed = 2024,
                       n = 1, r = 2, n_core = 1,
                       diag_less = 1e-2)
# will generate a warning because number of bootstraps is too small to
# estimate default confidence intervals
# Print results of the "BLB_archetypal" class object:
print(test)
```

```
# Summarize the "BLB_archetypal" class object:
summary(test)

}
```

`closer_grid_archetypal` *Performs the Archetypal Analysis of a Data Frame by using as Archetypes the Closer to The Archetypal Grid Data Points*

Description

the closer points to the archetypal grid are used as archetypes and then every data point is being expressed as a convex combination of those by using a modified PCHA method.

Usage

```
closer_grid_archetypal(dg,
                       diag_less = 1e-2,
                       niter=30,
                       use_seed = NULL,
                       verbose = TRUE)
```

Arguments

`dg`	The data frame with dimensions n x d
`diag_less`	The expected mean distance from 1 for the diagonal elements of submatrix A[irows,:], where `irows` are the closer to the `Grid Archetypal` data rows.
`niter`	The number of iterations that the A-update step should be done.
`use_seed`	If it is not NULL, then is used at the set.seed() for reproducibility reasons
`verbose`	If it is set to TRUE, then both initialization and iteration details are printed out

Details

The archetypal grid is being computed by taking the expand grid of the [Ximin,Ximax], i=1,...,d of all available variables. Then distances of all data points from that grid are calculated and the closer set of vectors is chosen.

Value

An object of class closer_grid_archetypal which is a list with members:

1. `grid`, the archetypal grid
2. `grid_rows`, the rows of the data frame that formed the archetypal grid
3. `aa`, an object of class archetypal

See Also

grid_archetypal

Examples

```
# Load package
library(GeomArchetypal)
# Create random data
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Closer Grid Archetypal
cga=closer_grid_archetypal(df,
             diag_less = 1e-2,
             niter = 150,
             verbose = FALSE)
# Print the class "closer_grid_archetypal":
print(cga)
# Summary of the class "closer_grid_archetypal":
summary(cga)
# Plot the class "closer_grid_archetypal":
plot(cga)
# Observe the Closer Grid Archetypes near the 8 corners of the cube ...
```

---

fast_archetypal *Performs the Archetypal Analysis of a Data Frame by using a Given Set of Archetypes*

Description

Performs the archetypal analysis of a data frame by using a known set of archetypes as rows of the data matrix.

Usage

```
fast_archetypal(df,
                irows,
                diag_less = 1e-2,
                niter = 30,
                verbose = TRUE,
                data_tables = TRUE,
                use_seed = NULL)
```

Arguments

df	The data frame with dimensions n x d
irows	The rows from data frame that represent the archetypes

diag_less	The expected mean distance from 1 for the diagonal elements of submatrix A[irows,:]
niter	The number of times that the A-update process should be done
verbose	If it is set to TRUE, then both initialization and iteration details are printed out
data_tables	If it set to TRUE, then a data table for the initial data points will be computed
use_seed	If it is not NULL, then is used at the set.seed() for reproducibility reasons

Details

If we know the archetypes, then we can bypass the half part of PCHA and perform only the A-update part, that of computing the convex combinations for each data point. Then archetypal analysis is a fast procedure, since we need only to compute one matrix.

Value

An object of class 'archetypal' is returned.

See Also

grid_archetypal, closer_grid_archetypal

Examples

```
# Load package
library(GeomArchetypal)
# Create random data
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Closer Grid Archetypal
cga=closer_grid_archetypal(df, diag_less = 1e-3,
                           niter = 250, verbose = FALSE)
# The closer to the Grid Archetypes points - rows are:
crows = cga$grid_rows
print(crows)
# Now we call the fast_archetypal() with those rows as argument:
fa=fast_archetypal(df, irows = crows, diag_less = 1e-3,
                   niter = 250, verbose = FALSE)
# Print:
print(fa)
# Summary:
summary(fa)
# Plot:
plot(fa)
# Results are identical to the closer_grid_archetypal() ones:
all.equal(cga$aa$BY,fa$BY)
```

`gallupGPS6` *Gallup Global Preferences Study processed data set of six variables*

Description

A 76132 x 6 data frame derived from Gallup Global Preferences Study, see [1] and [2] for details. It can be used as a big data set example.

Usage

```
data("gallupGPS6")
```

Format

A data frame with 76132 complete observations on the following 6 variables.

`patience` a numeric vector

`risktaking` a numeric vector

`posrecip` a numeric vector

`negrecip` a numeric vector

`altruism` a numeric vector

`trust` a numeric vector

Details

Data processing:

1. The non complete rows have been removed
2. The duplicated rows have also been removed

Note

1. The data was provided under a Creative Commons NonCommerical ShareAlike 4.0 license: https://creativecommons.org/licenses/by-nc-sa/4.0/
2. Other variables and identifiers from the original data have been dropped

Source

Individual data set was downloaded from https://www.gallup.com/analytics/318923/world-poll-public-datasets.aspx, last accessed 2024-03-09.

References

[1] Falk, A., Becker, A., Dohmen, T., Enke, B., Huffman, D., & Sunde, U. (2018). Global evidence on economic preferences. Quarterly Journal of Economics, 133 (4), 1645-1692.

[2] Falk, A., Becker, A., Dohmen, T. J., Huffman, D., & Sunde, U. (2016). The preference survey module: A validated instrument for measuring risk, time, and social preferences. IZA Discussion Paper No. 9674.

Examples

```
# Load package
library(GeomArchetypal)
data(gallupGPS6)
summary(gallupGPS6)
```

`grid_archetypal` *Performs the Archetypal Analysis of a Data Frame by using as Archetypes the Archetypal Grid*

Description

The archetypal grid is the expand grid of all intervals (X.imin,X.imax), i=1,...,d for a d-dimensional data frame.
That grid is used as archetypes and then only the A-update part of PCHA algorithm is used for computing the compositions of all data points.
The number of archetypes is always kappas = 2^d.

Usage

```
grid_archetypal(dg,
                diag_less = 1e-2,
                niter = 30,
                use_seed = NULL,
                verbose = TRUE)
```

Arguments

`dg`	The data frame with dimensions n x d
`diag_less`	The expected mean distance from 1 for the diagonal elements of submatrix A[1:kappas,:]
`niter`	The number of times that the A-update process should be done
`use_seed`	If it is not NULL, then is used at the set.seed() for reproducibility reasons
`verbose`	If it is set to TRUE, then both initialization and iteration details are printed out

Details

The archetypal grid defines a hyper-volume which contains the 100 % of all data points, if we take those grid points as the Convex Hull of all points. Although the archetypal grid points do not necessarily lie inside the data frame, here we do not care about that property: we only seek for the matrix of convex combinations (or composition matrix) A.

Value

An object of class grid_archetypal which is a list with members:

1. `grid`, the archetypal grid
2. `aa`, an object of class 'archetypal' which includes the archetypal grid as the first 2^d rows
3. `A`, the A-matrix with dimensions n x d that defines the compositions of all data points
4. `Y`, the matrix of initial data points

See Also

`closer_grid_archetypal`

Examples

```
# Load package
library(GeomArchetypal)
# Create random data:
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Grid Archetypal:
gaa=grid_archetypal(df, diag_less = 1e-6,
                    niter = 70, verbose = FALSE)
# Print class "grid_archetypal":
gaa
# Summary class "grid_archetypal":
summary(gaa)
# Plot class "grid_archetypal":
plot(gaa)
# Observe the Grid Archetypes at the 8 corners of the cube ..
```

---

plot.closer_grid_archetypal

Plot an Object of the Class closer_grid_archetypal

Description

It plots the output of `closer_grid_archetypal`

Usage

```
## S3 method for class 'closer_grid_archetypal'
plot(x, ...)
```

Arguments

x	An object of the class closer_grid_archetypal
...	Other arguments (ignored)

Details

Given an object of class closer_grid_archetypal the archetypal analysis result is plotted.

Value

No return value, called for side effects

Examples

```
# Load package
library(GeomArchetypal)
# Create random data
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Closer Grid Archetypal
cga=closer_grid_archetypal(df, niter = 70, verbose = FALSE)
# Plot the class "closer_grid_archetypal":
plot(cga)
```

---

plot.grid_archetypal *Plot an Object of the class grid_archetypal*

Description

It plots the output of grid_archetypal

Usage

```
## S3 method for class 'grid_archetypal'
plot(x, ...)
```

Arguments

x	An object of the class grid_archetypal
...	Other arguments (ignored)

Details

Given an object of class grid_archetypal the archetypal analysis result is plotted.
Remark: the first 2^d rows of the input data frame has Grid Archetypes (d is the dimension of the data points).

Value

No return value, called for side effects

Examples

```
# Load package
library(GeomArchetypal)
# Create random data:
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Grid Archetypal:
gaa=grid_archetypal(df, niter = 70, verbose = FALSE)
# Plot the class "archetypal":
plot(gaa)
```

---

points_inside_convex_hull

Computes the Percentage of Points that Lie Inside the Convex Hull which is Created by a Set of Vectors

---

Description

Given a set of k d-dimensional vectors which creates a Convex Hull (CH) we want to find the percentage of the n points of the n x d data frame df that lie inside that CH.

Usage

```
points_inside_convex_hull(df, dp)
```

Arguments

df	The n x d data frame of all available data points
dp	The k x d data frame of the given set of points that creates the Convex Hull

Details

In order for a really Convex Hull creation it must hold that: k >= d + 1, otherwise the problem is not well stated.

Value

A numeric output with percentage in two decimal digits

Note

Keep in mind that working with dimension greater than 6 will practical lead to extreme time executions. It highly suggested to work only for spaces with d<=6.

Author(s)

Demetris T. Christopoulos

Examples

```
# Load package
library(GeomArchetypal)
# Create random data:
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Grid Archetypal:
gaa=grid_archetypal(df, niter = 70, verbose = FALSE)
pc1=points_inside_convex_hull(df,gaa$grid)
print(pc1)
# [1] 100
# Closer Grid Archetypal:
cga=closer_grid_archetypal(df, niter = 70, verbose = FALSE)
pc2=points_inside_convex_hull(df,cga$aa$BY)
print(pc2)
# [1] 59
```

---

print.BLB_archetypal *Print an Object of the Class BLB_archetypal*

Description

It prints the output of BLB_archetypal

Usage

```
## S3 method for class 'BLB_archetypal'
print(x, ...)
```

Arguments

x	An object of the class BLB_archetypal
...	Other arguments (ignored)

Details

Given an object of class BLB_archetypal all the results are printed in explanatory form.

Value

No return value, called for side effects

Examples

```
{
library(GeomArchetypal)
library(mirai)
library(parallel)
data("gallupGPS6")
aa_var <- c("patience","risktaking","trust") # variables for archetypal analysis
# define the 2^3 archetypes from minimums and maximums of the data
min_var <- apply(gallupGPS6[, aa_var], 2, min, na.rm = TRUE)
max_var <- apply(gallupGPS6[, aa_var], 2, max, na.rm = TRUE)
temp <- as.data.frame.matrix(rbind(min_var, max_var))
colnames(temp) <- aa_var
list_minmax <- apply(temp, 2, as.list)
rm(temp, min_var, max_var)
arches <- data.matrix(do.call(expand.grid, list_minmax))
arches <- as.data.frame(arches)
rm(list_minmax)
# apply BLB archetypal for a minimal example
test <- BLB_archetypal(df = gallupGPS6,
                       ss_size = 50,
                       bs_size = nrow(gallupGPS6),
                       arches = arches,
                       use_seed = 2024,
                       n = 1, r = 2, n_core = 1,
                       diag_less = 1e-2)
# Print the results of the class "BLB_archetypal" object:
print(test)

}
```

---

print.closer_grid_archetypal

Print an Object of the closer_grid_archetypal

Description

It prints the output of closer_grid_archetypal

Usage

```
## S3 method for class 'closer_grid_archetypal'
print(x, ...)
```

Arguments

x	An object of the class closer_grid_archetypal
...	Other arguments (ignored)

Details

Given an object of class closer_grid_archetypal all the results are printed in explanatory form.

Value

No return value, called for side effects

Examples

```
# Load package
library(GeomArchetypal)
# Create random data
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Closer Grid Archetypal
cga=closer_grid_archetypal(df, niter = 70, verbose = FALSE)
# Print the class "closer_grid_archetypal"
print(cga)
```

---

print.grid_archetypal *Print an Object of the Class grid_archetypal*

Description

It prints the output of grid_archetypal

Usage

```
## S3 method for class 'grid_archetypal'
print(x, ...)
```

Arguments

x	An object of the class grid_archetypal
...	Other arguments (ignored)

Details

Given an object of class grid_archetypal all the results are printed in explanatory form.

Value

No return value, called for side effects

Examples

```
  # Load package
  library(GeomArchetypal)
# Create random data:
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Grid Archetypal:
gaa=grid_archetypal(df, niter = 70, verbose = FALSE)
# Print the class "grid_archetypal":
print(gaa)
```

---

summary.BLB_archetypal *Summarize an Object of the Class BLB_archetypal*

Description

It summarizes the output of BLB_archetypal

Usage

```
## S3 method for class 'BLB_archetypal'
summary(object, ...)
```

Arguments

object	An object of the class BLB_archetypal
...	Other arguments (ignored)

Details

Given an object of class BLB_archetypal all the results are being summarized in explanatory form.

Value

No return value, called for side effects

Examples

```
{
library(GeomArchetypal)
library(mirai)
library(parallel)
data("gallupGPS6")
aa_var <- c("patience","risktaking","trust") # variables for archetypal analysis
# define the 2^3 archetypes from minimums and maximums of the data
min_var <- apply(gallupGPS6[, aa_var], 2, min, na.rm = TRUE)
```

```
max_var <- apply(gallupGPS6[, aa_var], 2, max, na.rm = TRUE)
temp <- as.data.frame.matrix(rbind(min_var, max_var))
colnames(temp) <- aa_var
list_minmax <- apply(temp, 2, as.list)
rm(temp, min_var, max_var)
arches <- data.matrix(do.call(expand.grid, list_minmax))
arches <- as.data.frame(arches)
rm(list_minmax)
# apply BLB archetypal for a minimal example
test <- BLB_archetypal(df = gallupGPS6,
                       ss_size = 50,
                       bs_size = nrow(gallupGPS6),
                       arches = arches,
                       use_seed = 2024,
                       n = 1, r = 2, n_core = 1,
                       diag_less = 1e-2)
# Summarize the results of the "BLB_archetypal" class object:
summary(test)

}
```

`summary.closer_grid_archetypal` *Summary of an Object of the Class closer_grid_archetypal*

Description

It gives a summary for the output of `closer_grid_archetypal`

Usage

```
## S3 method for class 'closer_grid_archetypal'
summary(object, ...)
```

Arguments

`object`	An object of the class closer_grid_archetypal
`...`	Other arguments (ignored)

Details

Given an object of class closer_grid_archetypal the summary of the archetypal analysis output is given.

Value

No return value, called for side effects

Examples

```
# Load package
  library(GeomArchetypal)
# Create random data
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
# Closer Grid Archetypal
cga=closer_grid_archetypal(df, niter = 70, verbose = FALSE)
# Summary of the class "closer_grid_archetypal":
summary(cga)
```

`summary.grid_archetypal` *Summary of an Object of the Class grid_archetypal*

Description

It gives a summary for the output of `grid_archetypal`

Usage

```
## S3 method for class 'grid_archetypal'
summary(object, ...)
```

Arguments

`object`	An object of the class grid_archetypal
`...`	Other arguments (ignored)

Details

Given an object of class grid_archetypal the summary of the archetypal analysis output is given. Remark: the first 2^d rows of the input data frame are the Grid Archetypes (d is the dimension of the data points).

Value

No return value, called for side effects

Examples

```
# Load package
library(GeomArchetypal)
# Create random data:
set.seed(20140519)
df=matrix(runif(90) , nrow = 30, ncol=3)
colnames(df)=c("x","y","z")
```

```
# Grid Archetypal:
gaa=grid_archetypal(df, niter = 70, verbose = FALSE)
# Summary of the class "grid_archetypal":
summary(gaa)
```

Index