Package ‘adproclus’

November 10, 2023

Title Additive Profile Clustering Algorithms
Version 1.0.2
Depends R (>= 3.1.0)
License GPL (>= 3)
Encoding UTF-8
LazyData true
Imports checkmate, corrplot, gtools, igraph, NMFN, qgraph, stats, withr
RoxygenNote 7.2.3
Collate 'adproclus-package.R' 'adproclus_classes.R' 'clustering.R' 'data.R' 'get_starts.R' 'utils.R' 'visualize.R'
Suggests testthat (>= 3.0.0)
Config/testthat/edition 3
URL https://github.com/henry-heppe/adproclus
BugReports https://github.com/henry-heppe/adproclus/issues
NeedsCompilation no
Author Henry Heppe [aut, cre, cph], Julian Rossbroich [aut], Jeffrey Durieux [aut], Tom Wilderjans [aut]
Maintainer Henry Heppe <heppe.henry@gmail.com>
Repository CRAN
Date/Publication 2023-11-09 23:40:05 UTC
R topics documented:

- adpc
- adproclus
- adproclus_low_dim
- CGdata
- get_random
- get_rational
- get_semirandom
- plot.adpc
- plot_cluster_network
- plot_profiles
- plot_vers_by_comp
- print.adpc
- print.summary.adpc
- summary.adpc
- Summary

Index

adpc Constructor for a (low dimensional) ADPROCLUS solution object

Description

Yields an object of class adpc, which can be printed, plotted and summarized by the corresponding methods. Mandatory input are the membership matrix \( A \) and the profile matrix \( P \) (where the number of columns from \( A \) corresponds to the number of rows in \( P \)), if the object is to represent a full dimensional ADPROCLUS model. For a low dimensional ADPROCLUS model, the matrices \( C \) and \( B \) have to be provided and \( P \) can be inferred from those. All other inputs are optional but may be included so that the output from the summary(), print(), plot() is complete. For further details on the (low dimensional) ADPROCLUS model and what every element of the objects means see adproclus and adproclus_low_dim.

Usage

adpc(
A,
P,
sse = NULL,
totvar = NULL,
explvar = NULL,
iterations = NULL,
timer = NULL,
timer_one_run = NULL,
initial_start = NULL,
C = NULL,
B = NULL,
runs = NULL,
)
Arguments

A  Membership matrix A.
P  Profile matrix P.
sse  Sum of Squared Error.
totvar  Total variance.
explvar  Explained variance.
iterations  Number of iterations.
timer  Time needed to run the complete algorithm.
timer_one_run  Time to complete this single algorithm start.
initial_start  List containing type of start and start_allocation matrix.
C  Low dimensional profiles matrix C.
B  Matrix of base vectors connecting low dimensional components with original variables B.
runs  List of suboptimal models.
parameters  List of algorithm parameters.

Value

Object of class adpc.

Examples

# Create the information needed for a minimal object of class adpc
x <- stackloss
result <- adproclus(x, 3)
A <- result$A
P <- result$P

# Use constructor to obtain object of class adpc
result_object <- adpc(A, P)

---

adproclus  Additive profile clustering

Description

Perform additive profile clustering (ADPROCLUS) on object-by-variable data. Creates a model that assigns the objects to overlapping clusters which are characterized in terms of the variables by the so-called profiles.
Usage

adproclus(
  data,
  nclusters,
  start_allocation = NULL,
  nrandomstart = 3,
  nsemirandomstart = 3,
  algorithm = "ALS1",
  save_all_starts = FALSE,
  seed = NULL
)

Arguments

data Object-by-variable data matrix of class matrix or data.frame.
nclusters Number of clusters to be used. Must be a positive integer.
start_allocation Optional matrix of binary values as starting allocation for first run. Default is NULL.
nrandomstart Number of random starts (see get_random). Can be zero. Increase for better results, though longer computation time. Some research finds 500 starts to be a useful reference.
nsemirandomstart Number of semi-random starts (see get_semirandom). Can be zero. Increase for better results, though longer computation time. Some research finds 500 starts to be a useful reference.
algorithm Character string "ALS1" (default) or "ALS2", denoting the type of alternating least squares algorithm. Can be abbreviated with "1" or "2".
save_all_starts Logical. If TRUE, the results of all algorithm starts are returned. By default, only the best solution is retained.
seed Integer. Seed for the random number generator. Default: NULL, meaning no reproducibility.

Details

In this function, Mirkin’s (1987, 1990) Additive Profile Clustering (ADPROCLUS) method is used to obtain an unrestricted overlapping clustering model of the object by variable data provided by data.

The ADPROCLUS model approximates an \( I \times J \) object by variable data matrix \( X \) by an \( I \times J \) model matrix \( M \) that can be decomposed into an \( I \times K \) binary cluster membership matrix \( A \) and a \( K \times J \) real-valued cluster profile matrix \( P \), with \( K \) indicating the number of overlapping clusters. In particular, the aim of an ADPROCLUS analysis is therefore, given a number of clusters \( K \), to estimate a model matrix \( M = AP \) which reconstructs the data matrix \( X \) as close as possible in a least squares sense (i.e. sum of squared residuals). For a detailed illustration of the ADPROCLUS model and associated loss function, see Wilderjans et al. (2011).
The alternating least squares algorithms ("ALS1" and "ALS2") that can be used for minimization of the loss function were proposed by Depril et al. (2008). In "ALS2", starting from an initial random or rational estimate of \( A \) (see get_random and get_semirandom), \( A \) and \( P \) are alternately re-estimated conditionally upon each other until convergence. The "ALS1" algorithm differs from the previous one in that each row in \( A \) is updated independently and that the conditionally optimal \( P \) is recalculated after each row update, instead of the end of the matrix. For a discussion and comparison of the different algorithms, see Depril et al., 2008.

**Warning:** Computation time increases exponentially with increasing number of clusters, \( K \). We recommend to determine the computation time of a single start for each specific dataset and \( K \) before increasing the number of starts.

**Value**

adproclus() returns a list with the following components, which describe the best model (from the multiple starts):

- model matrix. The obtained overlapping clustering model \( M \) of the same size as data.
- \( A \) matrix. The membership matrix \( A \) of the clustering model. Clusters are sorted by size.
- \( P \) matrix. The profile matrix \( P \) of the clustering model.
- sse numeric. The residual sum of squares of the clustering model, which is minimized by the ALS algorithm.
- totvar numeric. The total sum of squares of data.
- explvar numeric. The proportion of variance in data that is accounted for by the clustering model.
- iterations numeric. The number of iterations of the algorithm.
- timer numeric. The amount of time (in seconds) the complete algorithm ran for.
- timer_one_run numeric. The amount of time (in seconds) the relevant single start ran for.
- initial_start list. Containing the initial membership matrix, as well as the type of start that was used to obtain the clustering solution. (as returned by get_random or get_semirandom)
- runs list. Each element represents one model obtained from one of the multiple starts. Each element contains all of the above information for the respective start.
- parameters list. Contains the parameters used for the model.

**References**


See Also

- `adproclus_low_dim` for low dimensional ADPROCLUS
- `get_random` for generating random starts
- `get_semirandom` for generating semi-random starts
- `get_rational` for generating rational starts

Examples

```r
# Loading a test dataset into the global environment
x <- stackloss

# Quick clustering with K = 2 clusters
clust <- adproclus(data = x, nclusters = 2)

# Clustering with K = 3 clusters,
# using the ALS2 algorithm,
# with 2 random and 2 semi-random starts
clust <- adproclus(x, 3,
                   nrandomstart = 2, nsemirandomstart = 2, algorithm = "ALS2"
)

# Saving the results of all starts
clust <- adproclus(x, 3,
                   nrandomstart = 2, nsemirandomstart = 2, save_all_starts = TRUE
)

# Clustering using a user-defined rational start profile matrix
# (here the first 4 rows of the data)
start <- get_rational(x, x[,1:4])$A
clust <- adproclus(x, 4, start_allocation = start)
```

---

**adproclus_low_dim**  
*Low dimensional ADPROCLUS*

### Description

Perform **low dimensional** additive profile clustering (ADPROCLUS) on object by variable data. Use case: data to cluster consists of a large set of variables, where it can be useful to interpret the cluster profiles in terms of a smaller set of components that represent the original variables well.

### Usage

```r
adproclus_low_dim(
  data,
  nclusters,
  ncomponents,
  start_allocation = NULL,
```
Arguments

- **data**: Object-by-variable data matrix of class matrix or data.frame.
- **nclusters**: Number of clusters to be used. Must be a positive integer.
- **ncomponents**: Number of components (dimensions) to which the profiles should be restricted. Must be a positive integer.
- **start_allocation**: Optional matrix of binary values as starting allocation for first run. Default is NULL.
- **nrandomstart**: Number of random starts (see `get_random`). Can be zero. Increase for better results, though longer computation time. Some research finds 500 starts to be a useful reference.
- **nsemirandomstart**: Number of semi-random starts (see `get_semirandom`). Can be zero. Increase for better results, though longer computation time. Some research finds 500 starts to be a useful reference.
- **save_all_starts**: logical. If TRUE, the results of all algorithm starts are returned. By default, only the best solution is retained.
- **seed**: Integer. Seed for the random number generator. Default: NULL, meaning no reproducibility.

Details

In this function, an extension by Depril et al. (2012) of Mirkins (1987, 1990) additive profile clustering method is used to obtain a low dimensional overlapping clustering model of the object by variable data provided by data. More precisely, the low dimensional ADPROCLUS model approximates an $I \times J$ object by variable data matrix $X$ by an $I \times J$ model matrix $M$. For $K$ overlapping clusters, $M$ can be decomposed into an $I \times K$ binary cluster membership matrix $A$ and a $K \times J$ real-valued cluster profile matrix $P$ s.t. $M = AP$. With the simultaneous dimension reduction, $P$ is restricted to be of reduced rank $S < \min(K, J)$, such that it can be decomposed into $P = CB'$, with $C$ a $K \times S$ matrix and $B$ a $J \times S$ matrix. Now, a row in $C$ represents the profile values associated with the respective cluster in terms of the $S$ components, while the entries of $B$ can be used to interpret the components in terms of the complete set of variables. In particular, the aim of an ADPROCLUS analysis is therefore, given a number of clusters $K$ and a number of dimensions $S$, to estimate a model matrix $M$ that reconstructs data matrix $X$ as close as possible in a least squares sense and simultaneously reduce the dimensions of the data. For a detailed illustration of the low dimensional ADPROCLUS model and associated loss function, see Depril et al. (2012).

**Warning**: Computation time increases exponentially with increasing number of clusters, $K$. We recommend to determine the computation time of a single start for each specific dataset and $K$ before increasing the number of starts.
Value

adproclus_low_dim() returns a list with the following components, which describe the best model (from the multiple starts):

- **model** matrix. The obtained overlapping clustering model $M$ of the same size as data.
- **model_lowdim** matrix. The obtained low dimensional clustering model $AC$ of size $I \times S$
- **A** matrix. The membership matrix $A$ of the clustering model. Clusters are sorted by size.
- **P** matrix. The profile matrix $P$ of the clustering model.
- **c** matrix. The profile values in terms of the low dimensional components.
- **sse** numeric. The residual sum of squares of the clustering model, which is minimized by the ALS algorithm.
- **totvar** numeric. The total sum of squares of data.
- **explvar** numeric. The proportion of variance in data that is accounted for by the clustering model.
- **iterations** numeric. The number of iterations of the algorithm.
- **timer** numeric. The amount of time (in seconds) the complete algorithm ran for.
- **timer_one_run** numeric. The amount of time (in seconds) the relevant single start ran for.
- **initial_start** list. A list containing the initial membership matrix, as well as the type of start that was used to obtain the clustering solution. (as returned by get_random or get_semirandom)
- **runs** list. Each element represents one model obtained from one of the multiple starts. Each element contains all of the above information.
- **parameters** list. Containing the parameters used for the model.

References


See Also

- adproclus for full dimensional ADPROCLUS
- get_random for generating random starts
- get_semirandom for generating semi-random starts
- get_rational for generating rational starts

Examples

# Loading a test dataset into the global environment
x <- stackloss

# Low dimensional clustering with K = 3 clusters
# where the resulting profiles can be characterized in S = 1 dimensions
clust <- adproclus_low_dim(x, 3, ncomponents = 1)
CGdata

Randomly generated data with underlying overlapping clusters.

Description
A computer generated object-by-variable dataset with an underlying nonrestricted overlapping clustering structure. For illustrative purposes within the ADPROCLUS package only.

Usage
CGdata

Format
A data frame with 100 rows and 15 variables

get_random

Generate initial random start

Description
Generate an initial random start for the (low dimensional) Additive Profile Clustering algorithm (see adproclus and adproclus_low_dim).

Usage
get_random(data, nclusters, seed = NULL)

Arguments
data Object-by-variable data matrix of class matrix or data.frame.
nclusters Number of clusters to be used. Must be a positive integer.
seed Integer. Seed for the random number generator. Default: NULL

Details
get_random generates a random initial binary membership matrix A such that each entry is an independent draw from a Bernoulli Distribution with \( \pi = 0.5 \).

For generating an initial start from random draws from the data, see get_semirandom. For generating an initial start based on a specific set of initial cluster centers, see get_rational.

Warning: This function does not obtain an ADPRCOLUS model. To perform additive profile clustering, see adproclus.
Value

get_random() returns a list with the following components:

- type  A character string denoting the type of start ('Random Start')
- A  A randomly generated initial Membership matrix

References


See Also

- adproclus, adproclus_low_dim for details about membership and profile matrices
- get_semirandom for generating semi-random starts
- get_rational for generating rational starts

Examples

```r
# Obtain data from data set "Stackloss" and generate start allocation
start_allocation <- get_random(stackloss, 3)$A
```

Description

If cluster profiles are given a priori, this function can be used to compute the conditionally optimal cluster membership matrix A which can then be used as a rational starting allocation for the (low dimensional) ADPROCLUS procedure (see adproclus and adproclus_low_dim).

Usage

get_rational(data, starting_profiles)

Arguments

data  Object-by-variable data matrix of class matrix or data.frame.

starting_profiles  A matrix where each row represents the profile values for a cluster. Needs to be of same dimensions as P.
**get_semirandom**

**Details**

The function uses the same quadratic loss function and minimization method as the (low dimensional) ADPROCLUS procedure does to find the next conditionally optimal membership matrix $A$. (for details, see Depril et al., 2012).

**Warning:** This function does *not* obtain an ADPRCOLUS model. To perform additive profile clustering, see `adproclus`.

**Value**

get_rational() returns a list with the following components:

- **type** - A character string denoting the type of start ('Rational Start')
- **$A** - An initial Membership matrix

**References**


**See Also**

- `adproclus`, `adproclus_low_dim` for details about membership and profile matrices
- `get_random` for generating random starts
- `get_semirandom` for generating semi-random starts

**Examples**

```r
# Obtain data from standard data set "Stackloss"
x <- stackloss

# Obtaining a user-defined rational start profile matrix
# (here the first 4 rows of the data)
start_allocation <- get_rational(x, x[1:4, ])$A
```

**Description**

Generate an initial semi-random start for the (low dimensional) Additive Profile Clustering algorithm (see `adproclus` and `adproclus_low_dim`).

**Usage**

get_semirandom(data, nclusters, seed = NULL)
get_semirandom

Arguments

data  Object-by-variable data matrix of class matrix or data.frame.
nclusters  Number of clusters to be used. Must be a positive integer.
seed  Integer. Seed for the random number generator. Default: NULL

Details

An initial cluster membership matrix $A$ is generated by finding the best $A$ conditional on an initial profile matrix $P$ generated by drawing $k$ randomly chosen, distinct, rows from data (for details, see Depril et al., 2012).

Warning: This function does not obtain an ADPRCLUS model. To perform additive profile clustering, see adproclus.

Value

get_semirandom returns a list with the following components:

type  A character string denoting the type of start (‘Semi-random Start’)
$A$  An initial Membership matrix

References


See Also

adproclus, adproclus_low_dim  for details about membership and profile matrices
get_random  for generating random starts
get_rational  for generating rational starts

Examples

# Obtain data from data set "Stackloss" and generate start allocation
start_allocation <- get_semirandom(stackloss, 3)$A
Description

When passing a (low dimensional) ADPROCLUS solution of class adpc to the generic plot(), this method plots the solution in one of the following three ways:

Network Each cluster is a vertex and the edge between two vertices represents the overlap between the corresponding clusters. The size of a vertex corresponds to the cluster size. The overlap is represented through color, width and numerical label of the edge. The numerical edge-labels can be relative (number of overlap observations / total observations) or absolute (number of observations in both clusters).

Profiles Plot the profile matrix ($P$ for full dimensional model, $C$ for low dimensional model) in the style of a correlation plot to visualize the relation of each cluster with each variable.

Variables by components Plot the low dimensional component-by-variable matrix $B'$ in the style of a correlation plot to visualize the relation of each component with each original variable.

NOTE: Only works for low dimensional ADPROCLUS.

Usage

## S3 method for class 'adpc'
plot(x, type = "Network", title = NULL, relative_overlap = TRUE, ...)

Arguments

x Object of class adpc. (Low dimensional) ADPROCLUS solution

type Choice for type of plot: one of "Network", "Profiles", "vars_by_comp". Default: "Network".

title String. OPTIONAL.

relative_overlap Logical, only applies to plot of type = "Network". If TRUE (default), the number of observations belonging to two clusters is divided by the total number of observations.

... additional arguments will be passed on to the functions plot_cluster_network(), plot_profiles(), plot_vars_by_comp()

Value

Invisibly returns the input model.
Examples

# Loading a test dataset into the global environment
x <- stackloss

# Quick low dimensional clustering with \( K = 3 \) clusters and \( S = 1 \) dimensions
clust <- adproclus_low_dim(x, 3, 1)

# Produce three plots of the model
plot(clust, type = "Network")
plot(clust, type = "Profiles")
plot(clust, type = "vars_by_comp")

Description

Produce a representation of a (low dimensional) ADPROCLUS solution, where each cluster is a vertex and the edge between two vertices represents the overlap between the corresponding clusters. The size of a vertex corresponds to the cluster size. The overlap is represented through color, width and numerical label of the edge. The numerical edge labels can be relative (number of overlap observations / total observations) or absolute (number of observations in both clusters). **NOTE:** This function can be called through the `plot(model, type = "Network")` function with model an object of class `adpc`.

Usage

```r
plot_cluster_network(
  model,
  title = "Cluster network of ADPROCLUS solution",
  relative_overlap = TRUE,
  filetype = NULL,
  filename = NULL,
  ...
)
```

Arguments

- **model**: ADPROCLUS solution (class: adpc). Low dimensional model possible.
- **title**: String. Default: "Cluster network of ADPROCLUS solution"
- **relative_overlap**: Logical. If `TRUE` (default), the number of observations belonging to two clusters is divided by the total number of observations. If `FALSE` the number of observations in a cluster overlap will be displayed on the edges.
### Description

Produce a representation of profile matrix $P$ (or $C$ for low dimensional solution) of an ADPROCLUS solution of class adpc. The plot displays the profiles in the style of a correlation plot. **NOTE:** This function can also be called through the `plot(model, type = "Profiles")` function with model an object of class adpc.

### Usage

```r
plot_profiles(model, title = "Profiles of ADPROCLUS solution", ...)  
```

### Arguments

- **model**: Object of class adpc. (Low dimensional) ADPROCLUS solution
- **title**: String. Default: "Profiles of ADPROCLUS solution"
- **...**: Additional arguments passing to the `corrplot::corrplot()` function, to customize the plot.

### Value

Invisibly returns the input model.
Examples

# Loading a test dataset into the global environment
x <- stackloss

# Quick clustering with K = 3 clusters
clust <- adproclus(x, 3)

# Plot the profile scores of each cluster
plot_profiles(clust)

---

plot_vars_by_comp  
Plot variable to component matrix of ADPROCLUS solution

Description

Produce a representation of variable to component matrix $B'$ of a low dimensional ADPROCLUS solution of class adpc. The plot displays the scores in the style of a correlation plot. **NOTE:** This function can be called through the `plot(model, type = "VarsByComp")` function with model an object of class adpc.

Usage

```r
plot_vars_by_comp(
  model,
  title = "B' of Low Dimensional ADPROCLUS Solution",
  ...
)
```

Arguments

- **model**: Object of class adpc. Must be **Low dimensional** ADPROCLUS solution
- **title**: String. Default: "B' of Low Dimensional ADPROCLUS Solution"
- **...**: Additional arguments passing to the `corrplot::corrplot()` function, to customize the plot

Value

Invisibly returns the input model.

Examples

# Loading a test dataset into the global environment
x <- stackloss

# Quick low dimensional clustering with K = 3 clusters and S = 1 dimensions
clust <- adproclus_low_dim(x, 3, 1)

# Plot the matrix B', connecting components with variables
plot_vars_by_comp(clust)
Description

For an object of class adpc as input, this method prints basic information about the ADPROCLUS solution represented by the object. Works for both full and low dimensional solutions. Adjust the parameters digits, matrix_rows, matrix_cols to change the level of detail printed.

Usage

```r
## S3 method for class 'adpc'
print(
  x,
  title = "ADPROCLUS solution",
  digits = 3,
  matrix_rows = 10,
  matrix_cols = 15,
  ...
)
```

Arguments

- `x`: ADPROCLUS solution (class: adpc)
- `title`: String. Default: "ADPROCLUS solution"
- `digits`: Integer. The number of digits that all decimal numbers will be rounded to.
- `matrix_rows`: Integer. The number of matrix rows to display. OPTIONAL
- `matrix_cols`: Integer. The number of matrix columns to display. OPTIONAL
- `...`: ignored

Value

No return value, called for side effects.

Examples

```r
# Obtain data, compute model, print model
x <- stackloss
model <- adproclus(x, 3)
print(model)
```
print.summary.adpc  

Print (low dimensional) ADPROCLUS summary

Description

Prints an object of class `summary.adpc` to represent and summarize a (low dimensional) ADPROCLUS solution. A number of parameters for how the results should be printed can be passed as an argument to `summary.adpc()` which then passes it on to this method. This method does not take a model of class `adpc` directly as input.

Usage

```r
## S3 method for class 'summary.adpc'
print(x, ...)
```

Arguments

- `x`: Object of class `summary.adpc`
- `...`: ignored

Value

Invisibly returns object of class `summary.adpc`.

Examples

```r
# Obtain data, compute model, print summary of model
x <- stackloss
model <- adproclus(x, 3)
print(summary(model))
```

summary.adpc  

Summary of ADPROCLUS solution

Description

For an object of class `adpc` as input, this method yields a summary object of class `summary.adpc` including group characteristics of the clusters in the solution. Works for both full and low dimensional solutions. Adjust the parameters `digits`, `matrix_rows`, `matrix_cols` to change the level of detail for the printing of the summary.
Usage

```r
## S3 method for class 'adpc'
summary(
  object,
  title = "ADPROCLUS solution",
  digits = 3,
  matrix_rows = 10,
  matrix_cols = 5,
  ...
)
```

Arguments

- **object** ADPROCLUS solution (class: adpc). Low dimensional model possible.
- **title** String. Default: "ADPROCLUS solution"
- **digits** Integer. The number of digits that all decimal numbers will be rounded to.
- **matrix_rows** Integer. The number of matrix rows to display. OPTIONAL
- **matrix_cols** Integer. The number of matrix columns to display. OPTIONAL
- **...** ignored

Value

Invisibly returns object of class `summary.adpc`.

Examples

```r
# Obtain data, compute model, summarize model
x <- stackloss
model <- adproclus(x, 3)
model_summary <- summary(model)
```
Index

* datasets
  CGdata, 9

adpc, 2
adproclus, 2, 3, 8–12
adproclus_low_dim, 2, 6, 6, 9–12

CGdata, 9

get_random, 4–8, 9, 11, 12
get_rational, 6, 8–10, 10, 12
get_semirandom, 4–11, 11

plot.adpc, 13
plot_cluster_network, 14
plot_profiles, 15
plot_vars_by_comp, 16
print.adpc, 17
print.summary.adpc, 18

summary.adpc, 18