Package 'DDAWASS'

November 4, 2026

Type Package

Title Discrete Distribution Data Analysis with Wasserstein Distance

Version 1.0.0

Date 2026-10-21

Description S4 classes and methods for the analysis of data described by discrete distributions with finite support (e.g. frequency distributions of ratings, scores or counts observed on groups of individuals), using the L2 Wasserstein metric between quantile functions. It is the discrete counterpart of 'HistDAWass' and provides exact Wasserstein distances (in C++), barycenters, Wasserstein variance, covariance and correlation, k-means and hierarchical clustering, quantile-based principal component and multiple factor analysis, two-component linear regression and reliability (Cronbach's alpha) indices. See Crocetta, Irpino, Antonucci and Marin (2025) <doi:10.1007/s11135-024-01956-0>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0), methods

Imports Rcpp (>= 1.0.0), HistDAWass, FactoMineR, ggplot2, rlang, tibble, stats, utils, grDevices

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), knitr, rmarkdown, CUB

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.1

URL https://github.com/Airpino/DDAWASS

BugReports https://github.com/Airpino/DDAWASS/issues

NeedsCompilation yes

Author Antonio Irpino [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-9293-7180>)

Maintainer Antonio Irpino <antonio.irpino@unicampania.it>

Repository CRAN

Date/Publication 2026-11-04 21:00:02 UTC

Contents

Arith-distributionD . . . 3
compQ-distributionD . . . 3
data2MatD . . . 4
distributionD . . . 5
distributionD-class . . . 6
dotpD . . . 6
get.MatD.stats . . . 7
long2MatD . . . 7
MatD . . . 8
MatD-accessors . . . 9
MatD-class . . . 10
MatD2long . . . 10
MatH2MatD . . . 11
plot,distributionD-method . . . 12
plot,MatD-method . . . 12
rQQ-distributionD . . . 13
Satisfaction . . . 14
simplifyD . . . 14
statsD . . . 15
tibble2MatD . . . 16
WassSqDistD . . . 17
WD.1d.PCA . . . 18
WD.bind . . . 19
WD.cronbach . . . 19
WD.MultiplePCA . . . 20
WD.plot.Spanish.fan . . . 21
WD.quantile.matrix . . . 21
WD.regression.GOF . . . 22
WD.regression.two.components . . . 23
WD.regression.two.components.predict . . . 24
WD.SSQ . . . 24
WD.var.covar . . . 25
WD.vec.mean . . . 26
WD.vec.mean.simplified . . . 26
WD_hclust . . . 27
WD_kmeans . . . 28
WD_MAT_DIST . . . 29
[,MatD,ANY,ANY,ANY-method . . . 30

Index **31**

Arith-distributionD *Arithmetic on discrete distributions*

Description

Operations are defined on quantile functions, consistently with the L2 Wasserstein geometry: the sum of two distributions is the distribution whose quantile function is $Q_1 + Q_2$; a distribution plus a number is a shift; a distribution times a number is a rescaling (a negative number also reverses the support, $Q(t) \to aQ(1-t)$).

Usage

```
## S4 method for signature 'distributionD,distributionD'
e1 + e2
```

Arguments

e1, e2	distributionD objects or numbers.

Value

a distributionD.

Examples

```
a <- distributionD(x = 1:3, p = c(0.2, 0.5, 0.3))
b <- distributionD(x = c(10, 20), p = c(0.5, 0.5))
a + b
a * 2 + 1
```

compQ-distributionD *Quantiles and cdf of a discrete distribution*

Description

Methods of the **HistDAWass** generics compQ and compP for distributionD objects. The quantile function is the left-continuous inverse of the cdf, $Q(t) = \min\{x : F(x) \geq t\}$, with $Q(0) = \min(x)$.

Usage

```
## S4 method for signature 'distributionD,ANY'
compQ(object, p)

## S4 method for signature 'distributionD,ANY'
compP(object, q)
```

Arguments

object	a distributionD.
p	a vector of probability levels in $[0, 1]$.
q	a vector of values.

Value

a numeric vector.

Examples

```
d <- distributionD(x = 1:5, p = c(0.05, 0.1, 0.2, 0.4, 0.25))
compQ(d, c(0, 0.25, 0.5, 0.75, 1))
compP(d, c(2, 3.5))
```

---

data2MatD *Build a MatD from raw (micro) data*

Description

Each row of data is an observation (e.g. a respondent) belonging to a group defined by one or more categorical columns (unit); for each group and each variable in vars the frequency distribution of the observed values is computed.

Usage

```
data2MatD(data, unit, vars, support = NULL, min.n = 1, sep = "_")
```

Arguments

data	a data.frame.
unit	name(s) or index(es) of the grouping column(s). With several columns, groups are their combinations (labels joined by sep); empty combinations are dropped.
vars	names or indices of the discrete numeric variables.
support	(optional) a list (one element per variable, or a single vector for all) with the admissible values (e.g. 1:7 for a 7-point Likert scale): values with zero frequency are then kept in the support, so that all the distributions of a variable share the same categories.
min.n	minimum number of observations for a group to be kept (groups with fewer rows are dropped with a message).
sep	separator used to build the labels of combined groups.

Value

a MatD with one row per group. The number of observations of each distribution is stored in its n slot (see get.MatD.stats(x, "n")).

Examples

```
set.seed(2)
raw <- data.frame(shop = rep(c("A", "B", "C"), each = 50),
                  gender = sample(c("F", "M"), 150, replace = TRUE),
                  q1 = sample(1:5, 150, replace = TRUE),
                  q2 = sample(0:10, 150, replace = TRUE))
data2MatD(raw, unit = "shop", vars = c("q1", "q2"))
data2MatD(raw, unit = c("shop", "gender"), vars = c("q1", "q2"),
          support = list(1:5, 0:10), min.n = 20)
```

`distributionD` *Constructor of a discrete distribution*

Description

Builds a `distributionD` object. Three ways to call it:

- `distributionD(x, p)`: support values and probabilities;
- `distributionD(x, counts = k)`: support values and absolute frequencies;
- `distributionD(obs)`: a vector of raw observations, which is tabulated.

Duplicated support values are merged, values with zero mass are dropped and the probabilities are normalised to sum to one.

Usage

```
distributionD(x = numeric(0), p = NULL, counts = NULL, keep.zeros = FALSE)
```

Arguments

`x`	numeric vector of support values, or of raw observations when both `p` and `counts` are `NULL`.
`p`	(optional) numeric vector of probabilities/relative frequencies.
`counts`	(optional) numeric vector of absolute frequencies.
`keep.zeros`	logical; keep support values with zero mass? Default `FALSE`.

Value

A `distributionD` object.

Examples

```
# from support and probabilities
d1 <- distributionD(x = 1:5, p = c(0.1, 0.2, 0.4, 0.2, 0.1))
# from counts
d2 <- distributionD(x = 0:10, counts = c(1, 0, 2, 3, 5, 10, 12, 20, 25, 15, 7))
# from raw observations
d3 <- distributionD(c(3, 4, 4, 5, 5, 5, 2, 4))
d1; meanD(d2); stdD(d3)
```

distributionD-class *Class distributionD*

Description

Class `distributionD` describes a discrete distribution with a finite support (e.g. the frequency distribution of a Likert/0-10 rating, of a count, or of any variable observed on a finite set of numeric values). It is the discrete counterpart of the `distributionH` class of **HistDAWass**.

Slots

`x` numeric vector: the support values, sorted in increasing order and without duplicates.

`p` numeric vector: the probability masses (relative frequencies) associated with `x`; they are positive and sum to one.

`cdf` numeric vector: the cumulative distribution function evaluated at `x` (`cumsum(p)`, last value equal to 1).

`n` numeric: the number of observations the distribution was computed from (`NA` when unknown).

`m` numeric: the mean of the distribution.

`s` numeric: the standard deviation of the distribution.

See Also

`distributionD` for the constructor.

dotpD *Inner product of quantile functions*

Description

`dotpD(a, b)` computes $\int_0^1 Q_a(t)Q_b(t)dt$, the building block of Wasserstein-based covariances.

Usage

```
dotpD(a, b)
```

Arguments

a, b	two `distributionD` objects.

Value

a number.

Examples

```
a <- distributionD(x = 1:3, p = c(0.2, 0.5, 0.3))
b <- distributionD(x = c(2, 4), p = c(0.5, 0.5))
dotpD(a, b)
```

---

get.MatD.stats *Statistics of the distributions in a MatD*

Description

Returns a matrix with a summary statistic computed for each cell.

Usage

```
get.MatD.stats(object, stat = "mean", prob = 0.5)
```

Arguments

object	a MatD.
stat	one of "mean", "median", "min", "max", "std", "var", "skewness", "kurtosis", "quantile", "n" (number of observations) or "k" (number of support values).
prob	probability level used when stat = "quantile".

Value

a list with elements stat and mat.

Examples

```
get.MatD.stats(Satisfaction)$mat[1:5, ]
get.MatD.stats(Satisfaction, stat = "quantile", prob = 0.1)$mat[1:5, ]
```

---

long2MatD *Build a MatD from a long frequency table*

Description

Build a MatD from a long frequency table

Usage

```
long2MatD(
  data,
  unit = "unit",
  variable = "variable",
  value = "value",
  freq = "freq"
)
```

Arguments

data	a data.frame in long format, with one row per (unit, variable, value) and a column of frequencies.
unit, variable, value, freq	names of the columns with the row labels, the variable names, the support values and the (absolute or relative) frequencies.

Value

a MatD.

Examples

```
tab <- expand.grid(unit = c("u1", "u2"), variable = c("A", "B"), value = 1:3)
tab$freq <- c(5, 1, 2, 2, 3, 3, 2, 2, 2, 6, 6, 6)
long2MatD(tab)
```

MatD *Constructor of a matrix of discrete distributions*

Description

Constructor of a matrix of discrete distributions

Usage

```
MatD(
  x = list(),
  nrows = 1,
  ncols = 1,
  rownames = NULL,
  varnames = NULL,
  by.row = FALSE
)
```

Arguments

x	a list of distributionD objects.
nrows	number of rows (units).
ncols	number of columns (variables).
rownames	(optional) names of the rows.
varnames	(optional) names of the columns.
by.row	logical; if TRUE the list x fills the matrix by rows, otherwise (default) by columns.

Value

A MatD object.

Examples

```
d <- list(distributionD(c(1, 2, 2, 3)), distributionD(c(2, 3, 3, 3)),
          distributionD(c(1, 1, 2)),    distributionD(c(3, 3, 2)))
MatD(d, nrows = 2, ncols = 2, rownames = c("A", "B"), varnames = c("Q1", "Q2"))
```

MatD-accessors *Accessors of a MatD object*

Description

Accessors of a MatD object

Usage

```
get.MatD.nrows(object)

get.MatD.ncols(object)

get.MatD.rownames(object)

get.MatD.varnames(object)

get.cell.MatD(object, r, c)

set.cell.MatD(object, r, c, value)
```

Arguments

object	a MatD object.
r, c	row and column index.
value	a distributionD.

Value

get.MatD.nrows and get.MatD.ncols return integers, get.MatD.rownames and get.MatD.varnames character vectors, get.cell.MatD a distributionD.

Examples

```
get.MatD.nrows(Satisfaction)
get.MatD.varnames(Satisfaction)
get.cell.MatD(Satisfaction, 1, 1)
```

MatD-class *Class MatD*

Description

Class MatD is a matrix whose cells are distributionD objects: rows are statistical units (individuals, groups, territories, ...) and columns are discrete distributional variables. It mirrors the MatH class of **HistDAWass**.

Slots

M a matrix of lists, each cell containing a distributionD.

MatD2long *Convert a MatD into a long data.frame*

Description

Convert a MatD into a long data.frame

Usage

```
MatD2long(object)
```

Arguments

object	a MatD.

Value

a data.frame with columns row, variable (factors, in the order of the MatD), x, p and cdf.

Examples

```
head(MatD2long(Satisfaction))
```

MatH2MatD *Discretize a matrix of histograms (MatH) into a MatD*

Description

Each histogram of a HistDAWass::MatH is transformed into a discrete distribution on n_levels classes: the masses are the probabilities of the classes defined by the breaks. This is the transformation used to compare continuous and discrete distributional data and to draw Green Eyes plots and distributional heatmaps of histogram data.

Usage

```
MatH2MatD(
  object,
  n_levels = 10,
  scheme = c("variable", "global", "standardized"),
  breaks = NULL,
  values = c("index", "midpoint"),
  return.breaks = FALSE
)
```

Arguments

object	a MatH.
n_levels	number of classes.
scheme	how the breaks are defined: "variable" (default, equal-width classes between the minimum and the maximum of each variable), "global" (the same classes for all variables) or "standardized" (each variable is standardised with its Wasserstein mean and standard deviation, and cut into equal-width classes between -3 and 3, with two open extreme classes).
breaks	(optional) a list with a vector of breaks for each variable (overrides scheme).
values	support values of the discrete distributions: "index" (default, the class number 1, ..., n_levels) or "midpoint" (the class centre in the original scale).
return.breaks	logical; if TRUE returns a list with the MatD and the breaks used.

Value

a MatD (or a list).

Examples

```
library(HistDAWass)
B <- MatH2MatD(BLOOD, n_levels = 10)
B
```

plot,distributionD-method *Plot a discrete distribution*

Description

Plot a discrete distribution

Usage

```
## S4 method for signature 'distributionD'
plot(x, y, type = "BAR", col = "#2E7D32", ...)
```

Arguments

x	a distributionD.
y	unused.
type	"BAR" (default, probability masses) or "CDF".
col	fill/line colour.
...	unused.

Value

a ggplot object.

Examples

```
d <- distributionD(x = 0:10, counts = c(1, 0, 2, 3, 5, 10, 12, 20, 25, 15, 7))
plot(d)
plot(d, type = "CDF")
```

plot,MatD-method *Plot a MatD*

Description

Draws a grid of bar charts (rows = units, columns = variables).

Usage

```
## S4 method for signature 'MatD'
plot(x, y, type = "BAR", col = "#2E7D32", ...)
```

Arguments

x	a MatD.
y	unused.
type	"BAR" (default) or "CDF".
col	fill colour.
...	unused.

Value

a ggplot object.

Examples

```
plot(Satisfaction[1:4, 1:4])
```

---

rQQ-distributionD *Quantile-quantile correlation of two discrete distributions*

Description

Method of the **HistDAWass** generic rQQ: the correlation between the quantile functions of two distributionD.

Usage

```
## S4 method for signature 'distributionD,distributionD'
rQQ(e1, e2)
```

Arguments

e1, e2	two distributionD objects.

Value

a number in $[-1, 1]$.

Examples

```
a <- distributionD(x = 1:5, p = c(0.05, 0.1, 0.2, 0.4, 0.25))
b <- distributionD(x = 1:5, p = c(0.3, 0.3, 0.2, 0.1, 0.1))
rQQ(a, b)
```

Satisfaction *Simulated hotel satisfaction survey*

Description

A MatD with 30 fictitious hotels (rows) and the distributions of the 0-10 scores given by their guests to seven items and to the overall satisfaction (columns). Data are simulated (see data-raw/Satisfaction.R in the source package); Satisfaction_raw contains the simulated individual answers.

Usage

```
Satisfaction
```

Format

Satisfaction: a MatD with 30 rows and 8 columns (Staff, Cleanliness, Comfort, Location, Price, Food, WiFi, Overall). Satisfaction_raw: a data.frame with one row per guest.

Examples

```
Satisfaction
plot(Satisfaction[1:3, 1:3])
```

simplifyD *Project discrete distributions on a fixed support*

Description

simplifyD moves the mass of each support value of a distribution to the nearest value of a given support (e.g. the categories 1:5 of a Likert scale): the value s_k receives the masses of all the points in $((s_{k-1}+s_k)/2, (s_k+s_{k+1})/2]$; masses below the first midpoint go to s_1 and masses above the last one to s_K. Ties (points exactly at a midpoint) go to the lower value.

Usage

```
simplifyD(object, support = NULL, keep.zeros = TRUE)
```

Arguments

object	a distributionD or a MatD.
support	the target support values. For a MatD it can be a list with one vector per column; if NULL (default, MatD only) each column is projected on the union of the support values observed in that column.
keep.zeros	logical; keep the support values that receive zero mass?

Details

The nearest-point rule gives the distribution supported on `support` closest to the original one in the L2 Wasserstein distance (indeed, in any Wasserstein distance): it is the Wasserstein projection on the set of distributions supported on `support`. It is typically used to express a barycenter (whose support is generally made of non-integer values) on the original categories of the variable, see `WD.vec.mean.simplified`. The mean of the projected distribution may differ from the original one, by at most half of the largest gap between consecutive support values.

For a barycenter $\bar{\mu}$ of $\mu_1, \ldots, \mu_n$, since $\sum_i w_i d_W^2(\mu_i, \nu) = \sum_i w_i d_W^2(\mu_i, \bar{\mu}) + d_W^2(\bar{\mu}, \nu)$, the projection of $\bar{\mu}$ on the support is also a barycenter of the data constrained to have that support. A proof is given in the file `barycenter_projection.pdf` installed with the package: `system.file("doc-theory", "barycenter_projection.pdf",package = "DDAWASS")`.

Value

an object of the same class as `object`.

Examples

```
b <- distributionD(x = c(1, 1.1, 1.35, 1.6, 1.8, 2.5, 2.6, 3.4, 4.9),
                   p = c(1, 2, 4, 8, 3, 5, 10, 6, 1))
s <- simplifyD(b, support = 1:5)
s
WassSqDistD(b, s)   # the smallest possible among distributions on 1:5
```

`statsD` *Basic statistics of a discrete distribution*

Description

`meanD`, `stdD`, `varD`, `skewD` and `kurtD` return the mean, the standard deviation, the variance, the (Pearson) skewness and the (non-excess) kurtosis of a `distributionD`.

Usage

```
meanD(object)

stdD(object)

varD(object)

skewD(object)

kurtD(object)

## S4 method for signature 'distributionD'
meanD(object)
```

```
## S4 method for signature 'distributionD'
stdD(object)

## S4 method for signature 'distributionD'
varD(object)

## S4 method for signature 'distributionD'
skewD(object)

## S4 method for signature 'distributionD'
kurtD(object)
```

Arguments

object	a distributionD object.

Value

a number.

Examples

```
d <- distributionD(x = 1:5, p = c(0.05, 0.1, 0.2, 0.4, 0.25))
meanD(d); stdD(d); skewD(d); kurtD(d)
```

---

tibble2MatD *Convert legacy nested tibbles into a MatD (and back)*

Description

The research code that preceded this package stored discrete distributional data as a tibble whose cells are data.frames with the support in the first column and a freq column (optionally n and cdf). tibble2MatD converts such objects into a MatD; MatD2tibble does the reverse.

Usage

```
tibble2MatD(tib, labels = NULL)

MatD2tibble(object)
```

Arguments

tib	a tibble/data.frame of list-columns.
labels	(optional) name or index of a character column holding the row labels; it is excluded from the variables.
object	a MatD.

Value

a MatD (tibble2MatD) or a tibble (MatD2tibble).

Examples

```
tb <- MatD2tibble(Satisfaction[1:3, 1:2])
tb
tibble2MatD(tb, labels = "ID_name")
```

WassSqDistD *L2 Wasserstein distance between discrete distributions*

Description

Computes the (squared) L2 Wasserstein distance between two distributionD objects, i.e. the L2 distance between their quantile functions. The computation is exact: quantile functions of discrete distributions are step functions and the integral is evaluated on the merged grid of cumulative probabilities.

Usage

```
WassSqDistD(object1, object2, ...)

## S4 method for signature 'distributionD,distributionD'
WassSqDistD(object1, object2, details = FALSE, ...)
```

Arguments

object1, object2	two distributionD objects.
...	unused.
details	logical; return the decomposition?

Details

With details = TRUE the squared distance is decomposed (Irpino and Romano, 2007) into a location, a size and a shape component:

$$d_W^2 = (\mu_1 - \mu_2)^2 + (\sigma_1 - \sigma_2)^2 + 2\sigma_1\sigma_2(1 - \rho_{QQ})$$

Value

the squared distance or, if details = TRUE, a named vector with the components SQ.W, POSITION, SIZE, SHAPE and rQQ.

References

Irpino A., Romano E. (2007). Optimal histogram representation of large data sets: Fisher vs piecewise linear approximation. *RNTI* E-9, 99–110.

Examples

```
a <- distributionD(x = 1:5, p = c(0.05, 0.1, 0.2, 0.4, 0.25))
b <- distributionD(x = 1:5, p = c(0.3, 0.3, 0.2, 0.1, 0.1))
WassSqDistD(a, b)
WassSqDistD(a, b, details = TRUE)
```

`WD.1d.PCA` *Principal component analysis of a single discrete distributional variable*

Description

Performs the quantile-based PCA of one distributional variable (Verde and Irpino, 2010): a classical non-standardised PCA (with `PCA`) of the matrix of quantiles, with the means and the standard deviations of the distributions as supplementary variables.

Usage

```
WD.1d.PCA(data, var = 1, quantiles = 10, plots = TRUE, ncp = 5)
```

Arguments

`data`	a `MatD`.
`var`	the column to analyse.
`quantiles`	number of quantile intervals (default 10).
`plots`	logical; draw the **FactoMineR** plots?
`ncp`	number of components kept.

Value

the `PCA` object of **FactoMineR** with the additional element `quantile_matrix`.

Examples

```
res <- WD.1d.PCA(Satisfaction, var = "Staff", plots = FALSE)
res$eig[1:3, ]
WD.plot.Spanish.fan(res)
```

WD.bind *Bind MatD objects by rows or by columns*

Description

Bind MatD objects by rows or by columns

Usage

```
WD.bind.col(a, b)

WD.bind.row(a, b)
```

Arguments

a, b	two MatD objects.

Value

a MatD.

Examples

```
WD.bind.col(Satisfaction[, 1:2], Satisfaction[, 3])
```

WD.cronbach *Cronbach's alpha for discrete distributional variables*

Description

Computes Cronbach's alpha using the Wasserstein covariance matrix of a set of distributional items (GLOB), and its decomposition into the part due to the covariance of the means (AVE) and the part due to the covariance of the centred distributions (VAR). The alpha.if.deleted matrix reports the values obtained removing each item in turn.

Usage

```
WD.cronbach(object)
```

Arguments

object	a MatD with at least three columns.

Value

a list with alpha and alpha.if.deleted.

Examples

```
WD.cronbach(Satisfaction)
```

`WD.MultiplePCA` *Multiple factor analysis of discrete distributional variables*

Description

Quantile-based multiple factor analysis (MFA) of several distributional variables: each variable is represented by a group of quantile columns and the groups are balanced as in `MFA`.

Usage

```
WD.MultiplePCA(
  data,
  list.of.vars,
  quantiles = 10,
  ncp = 5,
  pp = c(0, 1),
  plots = TRUE
)
```

Arguments

`data`	a `MatD`.
`list.of.vars`	the columns to analyse (at least two).
`quantiles`	number of quantile intervals (default 10).
`ncp`	number of components kept.
`pp`	range of probability levels.
`plots`	logical; draw the **FactoMineR** plots?

Value

the `MFA` object with the additional element `quantile_matrix`.

References

Crocetta C., Irpino A., Antonucci L., Marin C. (2025). A performance indicator and its decomposition according to the impacts of different aspects based on distributional data. *Quality & Quantity*, 59(3), 2091–2110. doi:10.1007/s11135024019560

Examples

```
res <- WD.MultiplePCA(Satisfaction, 1:4, quantiles = 5, plots = FALSE)
res$eig[1:3, ]
```

WD.plot.Spanish.fan *Spanish-fan plot of a quantile PCA*

Description

Draws the plot of the quantile variables of a WD.1d.PCA as consecutive coloured triangles (the "Spanish fan"): colours go from red (extreme quantiles) to yellow (median).

Usage

```
WD.plot.Spanish.fan(res, axes = c(1, 2))
```

Arguments

res	the output of WD.1d.PCA.
axes	the two axes to plot.

Value

a ggplot object.

Examples

```
res <- WD.1d.PCA(Satisfaction, var = "Price", plots = FALSE)
WD.plot.Spanish.fan(res, axes = c(1, 2))
```

WD.quantile.matrix *Matrix of quantiles of a MatD*

Description

For each row and each selected variable, the quantiles at levels seq(pp[1], pp[2], length.out = quantiles + 1) are computed and arranged in a numeric matrix (one block of quantiles + 1 columns per variable).

Usage

```
WD.quantile.matrix(
  object,
  quantiles = 10,
  vars = seq_len(ncol(object@M)),
  pp = c(0, 1)
)
```

Arguments

object	a MatD.
quantiles	number of quantile intervals.
vars	columns to use (default all).
pp	range of probability levels (default c(0, 1)).

Value

a numeric matrix.

Examples

```
WD.quantile.matrix(Satisfaction, quantiles = 4, vars = 1:2)[1:5, ]
```

WD.regression.GOF *Goodness of fit indices for distributional regression*

Description

Computes the root mean squared Wasserstein error (RMSE_W), the OMEGA index and the pseudo-R^2 of Irpino and Verde (2015), with the same definitions of HistDAWass::WH.regression.GOF.

Usage

```
WD.regression.GOF(observed, predicted)
```

Arguments

observed	a one-column MatD of observed distributions.
predicted	a one-column MatD of predicted distributions.

Value

a list with RMSE_W, OMEGA and PSEUDOR2.

Examples

```
xv <- c("Staff", "Cleanliness")
pars <- WD.regression.two.components(Satisfaction, "Overall", xv)
pred <- WD.regression.two.components.predict(Satisfaction[, xv], pars)
gof <- WD.regression.GOF(Satisfaction[, "Overall"], pred)
gof$OMEGA
```

`WD.regression.two.components`
Two-component regression of discrete distributional variables

Description

Fits the two-component linear model of Irpino and Verde (2015) to discrete distributional data:

$$Q_{y_i}(t) = \beta_0 + \sum_j \beta_j \mu_{x_{ij}} + \sum_j \gamma_j Q^c_{x_{ij}}(t) + e_i(t)$$

where Q^c is the centred quantile function ($Q - \mu$) and $\gamma_j \geq 0$ so that the predicted quantile functions are non-decreasing. Since centred quantile functions are orthogonal to constants, the least-squares problem splits into an OLS regression on the means (β) and a non-negative least-squares problem on the centred parts (γ), both solved exactly.

Usage

```
WD.regression.two.components(data, Yvar, Xvars)
```

Arguments

`data`	a `MatD`.
`Yvar`	the response column.
`Xvars`	the explanatory columns.

Value

a named vector of parameters: `(AV_Intercept)`, the `AV_` coefficients of the means and the `CEN_` coefficients of the centred distributions.

References

Irpino A., Verde R. (2015). Linear regression for numeric symbolic variables: a least squares approach based on Wasserstein distance. *Advances in Data Analysis and Classification*, 9(1), 81–106. doi:10.1007/s11634015019-77

Examples

```
pars <- WD.regression.two.components(Satisfaction, Yvar = "Overall",
                                     Xvars = c("Staff", "Price", "Cleanliness"))
pars
xv <- c("Staff", "Price", "Cleanliness")
pred <- WD.regression.two.components.predict(Satisfaction[, xv], pars)
WD.regression.GOF(Satisfaction[, "Overall"], pred)[c("RMSE_W", "OMEGA")]
```

WD.regression.two.components.predict *Predictions of the two-component regression*

Description

Predictions of the two-component regression

Usage

```
WD.regression.two.components.predict(data, parameters)
```

Arguments

data	a MatD with the explanatory variables (in the same order used for the estimation).
parameters	the output of WD.regression.two.components.

Value

a one-column MatD of predicted distributions.

Examples

```
xv <- c("Staff", "Price")
pars <- WD.regression.two.components(Satisfaction, "Overall", xv)
pred <- WD.regression.two.components.predict(Satisfaction[, xv], pars)
pred[1:3, ]
```

WD.SSQ *Total sum of squares of a discrete distributional variable*

Description

WD.SSQ returns the (weighted) sum of squared Wasserstein distances of the distributions of each column from their barycenter (the deviance).

Usage

```
WD.SSQ(object, w = 1)
```

Arguments

object	a MatD.
w	(optional) weights of the rows (not normalised).

Value

a named numeric vector (one value per column).

Examples

```
WD.SSQ(Satisfaction[, 1:3])
```

`WD.var.covar` *Wasserstein variance-covariance and correlation matrices*

Description

`WD.var.covar` computes the covariance matrix of the discrete distributional variables of a `MatD`, following the L2 Wasserstein (quantile) approach of Irpino and Verde (2015):

$$C_{jk} = \sum_i w_i \int_0^1 (Q_{ij}(t) - \bar{Q}_j(t))(Q_{ik}(t) - \bar{Q}_k(t))dt$$

where $\bar{Q}_j$ is the quantile function of the barycenter of variable j. The diagonal contains the Wasserstein (Frechet) variances. `WD.correlation` returns the corresponding correlation matrix.

Usage

```
WD.var.covar(object, w = 1)

WD.correlation(object, w = 1)
```

Arguments

`object`	a `MatD`.
`w`	(optional) weights of the rows.

Value

a p x p matrix.

References

Irpino A., Verde R. (2015). Basic statistics for distributional symbolic variables: a new metric-based approach. *Advances in Data Analysis and Classification*, 9(2), 143–175. doi:10.1007/s11634014-01764

Examples

```
WD.var.covar(Satisfaction)
round(WD.correlation(Satisfaction), 3)
```

WD.vec.mean *Barycenter (Frechet mean) of discrete distributional variables*

Description

Computes, for each column of a MatD, the (weighted) L2 Wasserstein barycenter of its distributions, i.e. the distribution whose quantile function is the weighted average of the quantile functions. The barycenter of discrete distributions is itself discrete; its support is in general not a subset of the original supports.

Usage

```
WD.vec.mean(object, w = 1)
```

Arguments

object	a MatD.
w	(optional) weights of the rows (default: equal weights).

Value

a distributionD if object has one column, otherwise a one-row MatD.

Examples

```
WD.vec.mean(Satisfaction[, "Staff"])
WD.vec.mean(Satisfaction)
```

WD.vec.mean.simplified *Barycenter expressed on the original support values*

Description

Computes the exact Wasserstein barycenter of each column of a MatD (see WD.vec.mean) and then projects it on the support of the variable with simplifyD, so that the result has the same categories of the data (e.g. the 1-7 scores of a Likert item) and new probabilities. The result is the distribution on the given support closest to the exact barycenter, and it is also the barycenter of the data constrained to that support (see simplifyD). It is an approximation, useful for presentation and interpretation: the exact barycenter, returned by WD.vec.mean, is the one used in all the computations of the package (variances, covariances, k-means prototypes, ...).

Usage

```
WD.vec.mean.simplified(object, w = 1, support = NULL)
```

Arguments

object	a MatD.
w	(optional) weights of the rows.
support	(optional) the target support (a vector, or a list with one vector per column); by default the union of the support values observed in each column.

Value

a distributionD if object has one column, otherwise a one-row MatD.

Examples

```
exact <- WD.vec.mean(Satisfaction[, "Staff"])
approx <- WD.vec.mean.simplified(Satisfaction[, "Staff"], support = 0:10)
approx
WassSqDistD(exact, approx)
```

WD_hclust *Hierarchical clustering of discrete distributional data*

Description

Agglomerative clustering (hclust) based on the L2 Wasserstein distance between rows.

Usage

```
WD_hclust(x, simplify = FALSE, method = "complete", weights = 1)
```

Arguments

x	a MatD.
simplify	ignored (compatibility with WH_hclust).
method	agglomeration method passed to hclust (default "complete"; "ward.D2" is the natural choice for the Wasserstein metric).
weights	(optional) weights of the variables.

Value

an object of class hclust.

Examples

```
hc <- WD_hclust(Satisfaction, method = "ward.D2")
plot(hc)
```

WD_kmeans *K-means clustering of discrete distributional data*

Description

Dynamic clustering (k-means) of the rows of a MatD using the L2 Wasserstein distance; prototypes are the Wasserstein barycenters of the clusters. Initial prototypes are chosen with the k-means++ seeding, the best of rep runs is returned.

Usage

```
WD_kmeans(
  x,
  k,
  rep = 5,
  simplify = FALSE,
  qua = 10,
  standardize = FALSE,
  weights = 1,
  iter.max = 100,
  verbose = FALSE
)
```

Arguments

x	a MatD.
k	number of clusters.
rep	number of repetitions (random starts).
simplify	ignored (compatibility with WH_kmeans).
qua	ignored (compatibility with WH_kmeans).
standardize	logical; if TRUE each variable is weighted by the inverse of its Wasserstein variance.
weights	(optional) weights of the variables (overridden by standardize).
iter.max	maximum number of iterations.
verbose	print progress?

Value

a list with solution (a list with IDX, the cluster labels, cardinality, centers, a MatD of prototypes, and Crit, the within sum of squares), quality (the ratio between the between and the total sum of squares), TSS and all.crit, the criteria of all the repetitions.

Examples

```
set.seed(1)
km <- WD_kmeans(Satisfaction, k = 3, rep = 5)
km$solution$IDX
km$quality
```

WD_MAT_DIST *Matrix of L2 Wasserstein distances between the rows of a MatD*

Description

The distance between two rows (units) is $d(i,i') = \sqrt{\sum_j \lambda_j d_W^2(x_{ij}, x_{i'j})}$. The computation is done in C++ with an exact linear-time algorithm.

Usage

```
WD_MAT_DIST(object, simplify = FALSE, weights = 1, squared = FALSE)
```

Arguments

object	a MatD.
simplify	ignored (kept for compatibility with WH_MAT_DIST).
weights	(optional) weights λ_j of the variables.
squared	logical: return squared distances? Default FALSE.

Value

a symmetric n x n matrix.

Examples

```
D <- WD_MAT_DIST(Satisfaction)
round(D[1:5, 1:5], 3)
```

`[,MatD,ANY,ANY,ANY-method`

Extract a sub-matrix of a MatD

Description

Extract a sub-matrix of a MatD

Usage

```
## S4 method for signature 'MatD,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	a `MatD`.
`i, j`	row and column indices (numbers, names or logicals).
`...`	unused.
`drop`	ignored: the result is always a `MatD`.

Value

a `MatD`.

Examples

```
Satisfaction[1:3, c("Staff", "Price")]
```

Index

∗ **datasets**
    Satisfaction, 14
*,distributionD,numeric-method
        *(*Arith-distributionD*)*, 3
*,numeric,distributionD-method
        *(*Arith-distributionD*)*, 3
+,distributionD,distributionD-method
        *(*Arith-distributionD*)*, 3
+,distributionD,numeric-method
        *(*Arith-distributionD*)*, 3
+,numeric,distributionD-method
        *(*Arith-distributionD*)*, 3
-,distributionD,numeric-method
        *(*Arith-distributionD*)*, 3
[,MatD,ANY,ANY,ANY-method, 30

Arith-distributionD, 3

compP,distributionD,ANY-method
        *(*compQ-distributionD*)*, 3
compP,distributionD-method
        *(*compQ-distributionD*)*, 3
compQ,distributionD,ANY-method
        *(*compQ-distributionD*)*, 3
compQ,distributionD-method
        *(*compQ-distributionD*)*, 3
compQ-distributionD, 3

data2MatD, 4
distributionD, *3*, *5*, 5, *6*, *10*, *13*, *15*, *17*
distributionD-class, 6
dotpD, 6

get.cell.MatD *(*MatD-accessors*)*, 9
get.MatD.ncols *(*MatD-accessors*)*, 9
get.MatD.nrows *(*MatD-accessors*)*, 9
get.MatD.rownames *(*MatD-accessors*)*, 9
get.MatD.stats, 7
get.MatD.varnames *(*MatD-accessors*)*, 9

hclust, *27*

kurtD *(*statsD*)*, 15
kurtD,distributionD-method *(*statsD*)*, 15

long2MatD, 7

MatD, 8, *9*, *14*, *26*, *28*
MatD-accessors, 9
MatD-class, 10
MatD2long, 10
MatD2tibble *(*tibble2MatD*)*, 16
MatH2MatD, 11
meanD *(*statsD*)*, 15
meanD,distributionD-method *(*statsD*)*, 15
MFA, *20*

PCA, *18*
plot,distributionD-method, 12
plot,MatD-method, 12

rQQ,distributionD,distributionD-method
        *(*rQQ-distributionD*)*, 13
rQQ-distributionD, 13

Satisfaction, 14
Satisfaction_raw *(*Satisfaction*)*, 14
set.cell.MatD *(*MatD-accessors*)*, 9
simplifyD, 14, *26*
skewD *(*statsD*)*, 15
skewD,distributionD-method *(*statsD*)*, 15
statsD, 15
stdD *(*statsD*)*, 15
stdD,distributionD-method *(*statsD*)*, 15

tibble2MatD, 16

varD *(*statsD*)*, 15
varD,distributionD-method *(*statsD*)*, 15

WassSqDistD, 17
WassSqDistD,distributionD,distributionD-method
        *(*WassSqDistD*)*, 17

WD.1d.PCA, 18, *21*
WD.bind, 19
WD.correlation *(*WD.var.covar*)*, 25
WD.cronbach, 19
WD.MultiplePCA, 20
WD.plot.Spanish.fan, 21
WD.quantile.matrix, 21
WD.regression.GOF, 22
WD.regression.two.components, 23, *24*
WD.regression.two.components.predict, 24
WD.SSQ, 24
WD.var.covar, 25
WD.vec.mean, *26*, 26
WD.vec.mean.simplified, *15*, 26
WD_hclust, 27
WD_kmeans, 28
WD_MAT_DIST, 29