Package: IntervalQuestionStat (via r-universe)

September 15, 2024

Title Tools to Deal with Interval-Valued Responses in Questionnaires

Type Package

Version 0.2.0

Description A user-friendly toolbox for doing the statistical analysis of interval-valued responses in questionnaires measuring intrinsically imprecise human attributes or features (attitudes, perceptions, opinions, feelings, etc.). In particular, this package provides S4 classes, methods, and functions in order to compute basic arithmetic and statistical operations with interval-valued data; prepare customized plots; associate each interval-valued response to its equivalent Likert-type and visual analogue scales answers through the minimum theta-distance and the mid-point criteria; analyze the reliability of respondents' answers from the internal consistency point of view by means of Cronbach's alpha coefficient; and simulate interval-valued responses in this type of questionnaires. The package also incorporates some real-life data that can be used to illustrate its working with several non-trivial reproducible examples. The methodology used in this package is based in many theoretical and applied publications from SMIRE+CoDiRE (Statistical Methods with Imprecise Random Elements and Comparison of Distributions of Random Elements) Research Group (<https://bellman.ciencias.uniovi.es/smire+codire/>) from the University of Oviedo (Spain).

License LGPL (>= 3)

Author José García-García [aut, cre] (<https://orcid.org/0000-0003-0991-1866>), María Asunción Lubiano [ctb] (<https://orcid.org/0000-0002-9847-5164>)

Maintainer José García-García <garciagarjose@uniovi.es>

URL https://github.com/garciagarjose/IntervalQuestionStat/

BugReports https://github.com/garciagarjose/IntervalQuestionStat/issues/

Encoding UTF-8

NeedsCompilation no

LazyData true

Depends R (>= 3.5.0)

Imports methods, grDevices, graphics, stats

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Suggests testthat (>= 3.0.0), vdiffr

Config/testthat/edition 3

Repository https://garciagarjose.r-universe.dev

RemoteUrl https://github.com/garciagarjose/intervalquestionstat

RemoteRef HEAD

RemoteSha e2c05b4689746d8bc6b17154695b10f0d762d29a

Contents

`IntervalQuestionStat-package`

Tools to Deal with Interval-Valued Responses in Questionnaires

---

Description

IntervalQuestonStat is an open source package for doing the statistical analysis of interval-valued responses collected through interval-valued scales in lots of different widely used questionnaires measuring many intrinsically imprecise human attributes in the R environment. In particular, this package implements the theoretical concepts, results, and ideas suggested by the SMIRE+CoDiRE (*Statistical Methods with Imprecise Random Elements and Comparison of Distributions of Random Elements*) Research Group (`https://bellman.ciencias.uniovi.es/smire+codire/`) from the University of Oviedo (Spain) taking into account some applied investigations and real-life studies.

Details

In Social and Educational Sciences and many other disciplines, interval-valued scales arise as a strong alternative to both traditional Likert-type or visual analogue scales in some questionnaires measuring people's behavior (attitudes, opinions, perceptions, feelings, etc.). This type of data can not be directly measured because it concerns inherently imprecise features. Likert-type and visual analogue scales force respondents to choose single-point answers linked to some items (statements or questions), so individual differences are almost systematically overlooked. In order to overcome the limitations of these scales in capturing uncertainty over respondents' answers, interval-valued scales allow them to select a real-valued interval and not being constrained to a single point.

This package provides S4 classes, methods, and functions for dealing with this type of data and it also includes some real-life data sets. In particular, it aims to provide the following functionality:

1. Definition of interval-valued objects and instances (see IntervalData-class, `IntervalData`, IntervalList-class, `IntervalList`, IntervalMatrix-class, and `IntervalMatrix`).
2. Calculation of basic operations with interval-valued data (see arithmetic and `distance`).
3. Calculation of some central tendency and variation measures (see `mean`, `var` and `cov`).
4. Visualization of interval-valued data (see `plot`).
5. Association of interval-valued responses and their corresponding equivalent Likert-type and visual analogue scales responses (see `ivs2likert` and `ivs2vas`).
6. Statistical analysis of reliability of questionnaire's responses (see `cronbach`).
7. Simulation of interval-valued responses in questionnaires (see `simulIVS`).

For a complete list of classes, methods and functions included in the **IntervalQuestionStat** package call `help(package="IntervalQuestionStat")` on the R console.

Acknowledgments: The initial development of this R package has been partially supported by the Principality of Asturias Grant AYUD/2021/50897 and also by the Spanish Ministry of Economy and Business Grant PID2019-104486GB-I00.

Author(s)

José García-García <garciagarjose@uniovi.es>,
with contributions from María Asunción Lubiano <lubiano@uniovi.es>.

References

- Aumann, R.J. (1965). Integrals of set-valued functions. *Journal of Mathematical Analysis and Applications*, 12(1):1-12. doi:10.1016/0022247X(65)900491.
- Cronbach L.J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16, 297-334. doi:1001007/BF02310555.
- De la Rosa de Sáa, S.; Gil, M.Á.; González-Rodríguez, G.; López, M.T.; Lubiano M.A. (2015). Fuzzy rating scale-based questionnaires and their statistical analysis, *IEEE Transactions on Fuzzy Systems*, 23(1):111-126. doi:10.1109/TFUZZ.2014.2307895.
- Fréchet, M. (1948). Lés éléments aléatoires de nature quelconque dans un espace distancié. *Annales de l'institut Henri Poincaré*, 10(4):215-310.
- Gil, M.Á.; Lubiano, M.A.; Montenegro, M.; López, M.T. (2002). Least squares fitting of an affine function and strength of association for interval-valued data. *Metrika*, 56:97-111. doi:10.1007/s001840100160.
- Hankin, R.K.S. (2010).. A step-by-step guide to writing a simple package that uses S4 methods: a "hello world" example. Technical Report. Auckland University of Technology.
- Lubiano, M.A.; García-Izquierdo, A.L.; Gil, M.Á. (2021). Fuzzy rating scales: Does internal consistency of a measurement scale benefit from coping with imprecision and individual differences in psychological rating?. *Information Sciences*, 550:91-108. doi:10.1016/j.ins.2020.10.042.
- Minkowski, H. (1903). Volumen und oberfläche. *Mathematische Annalen*, 57:447-495.
- Moore, R.E.; Kearfott, R.B.; Cloud, M.J. (2009). Introduction to Interval Analysis. *Society for Industrial and Applied Mathematics*, USA. doi:10.1137/1.9780898717716.

`apply` *Apply functions over* `IntervalMatrix` *margins*

Description

This function allows to apply a function over the rows or columns of an interval-valued matrix.

Usage

```
## S4 method for signature 'IntervalMatrix'
apply(X, MARGIN, FUN)
```

Arguments

X	A matrix of interval-valued data stored as an IntervalMatrix object.
MARGIN	A single numeric value giving the direction which the function will be applied over. In this case, only two different options are allowed: • 1: The function will be applied by rows. • 2: The function will be applied by columns.
FUN	The function to be applied over the selected interval-valued matrix margins.

Value

This function returns the numeric vector or the list of interval-valued data attained by applying the selected function to the specified margin (rows or columns) of an interval-valued matrix. Therefore, this function always returns either a numeric or either an IntervalList object, respectively.

Author(s)

José García-García <garciagarjose@uniovi.es>

Examples

```
## Some apply() examples
## IntervalMatrix definition
m <- IntervalMatrix(matrix(c(0, 1, 2, 3, 0, 3, 4, 9), 2, 4))
## Calculate sample Aumann means by rows and columns
## These code lines return IntervalList objects
apply(m, 1, mean)
apply(m, 2, mean)
## Calculate sample Fréchet variance by rows and columns
## These code lines return numeric vectors
apply(m, 1, var)
apply(m, 2, var)
```

---

arithmetic *Interval arithmetic operations*

Description

These commands apply basic interval-valued arithmetic operations.

Usage

```
## S4 method for signature 'numeric,IntervalData'
e1 + e2
## S4 method for signature 'IntervalData,numeric'
e1 + e2
## S4 method for signature 'IntervalData,IntervalData'
e1 + e2
## S4 method for signature 'numeric,IntervalData'
e1 - e2
## S4 method for signature 'IntervalData,numeric'
e1 - e2
## S4 method for signature 'IntervalData,IntervalData'
e1 - e2
## S4 method for signature 'IntervalData,ANY'
e1 - e2 # -e1
## S4 method for signature 'numeric,IntervalData'
e1 * e2
## S4 method for signature 'IntervalData,numeric'
e1 * e2

## S4 method for signature 'numeric,IntervalList'
e1 + e2
## S4 method for signature 'IntervalList,numeric'
e1 + e2
## S4 method for signature 'IntervalData,IntervalList'
e1 + e2
## S4 method for signature 'IntervalList,IntervalData'
e1 + e2
## S4 method for signature 'IntervalList,IntervalList'
e1 + e2
## S4 method for signature 'numeric,IntervalList'
e1 - e2
## S4 method for signature 'IntervalList,numeric'
e1 - e2
## S4 method for signature 'IntervalData,IntervalList'
e1 - e2
## S4 method for signature 'IntervalList,IntervalData'
e1 - e2
## S4 method for signature 'IntervalList,IntervalList'
e1 - e2
## S4 method for signature 'IntervalList,ANY'
e1 - e2 # -e1
## S4 method for signature 'numeric,IntervalList'
e1 * e2
## S4 method for signature 'IntervalList,numeric'
e1 * e2

## S4 method for signature 'numeric,IntervalMatrix'
```

```
e1 + e2
## S4 method for signature 'IntervalMatrix,numeric'
e1 + e2
## S4 method for signature 'matrix,IntervalMatrix'
e1 + e2
## S4 method for signature 'IntervalMatrix,matrix'
e1 + e2
## S4 method for signature 'IntervalData,IntervalMatrix'
e1 + e2
## S4 method for signature 'IntervalMatrix,IntervalData'
e1 + e2
## S4 method for signature 'IntervalList,IntervalMatrix'
e1 + e2
## S4 method for signature 'IntervalMatrix,IntervalList'
e1 + e2
## S4 method for signature 'IntervalMatrix,IntervalMatrix'
e1 + e2
## S4 method for signature 'numeric,IntervalMatrix'
e1 - e2
## S4 method for signature 'IntervalMatrix,numeric'
e1 - e2
## S4 method for signature 'matrix,IntervalMatrix'
e1 - e2
## S4 method for signature 'IntervalMatrix,matrix'
e1 - e2
## S4 method for signature 'IntervalData,IntervalMatrix'
e1 - e2
## S4 method for signature 'IntervalMatrix,IntervalData'
e1 - e2
## S4 method for signature 'IntervalList,IntervalMatrix'
e1 - e2
## S4 method for signature 'IntervalMatrix,IntervalList'
e1 - e2
## S4 method for signature 'IntervalMatrix,IntervalMatrix'
e1 - e2
## S4 method for signature 'IntervalMatrix,ANY'
e1 - e2 # -e1
## S4 method for signature 'numeric,IntervalMatrix'
e1 * e2
## S4 method for signature 'IntervalMatrix,numeric'
e1 * e2
## S4 method for signature 'matrix,IntervalMatrix'
e1 * e2
## S4 method for signature 'IntervalMatrix,matrix'
e1 * e2
```

Arguments

e1	A single numeric value, a vector, a matrix, a single interval, or a list or a matrix of intervals saved as a `numeric`, `matrix`, `IntervalData`, `IntervalList`, or `IntervalMatrix` object, respectively.
e2	A single numeric value, a vector, a matrix, a single interval, or a list or a matrix of intervals saved as a `numeric`, `matrix`, `IntervalData`, `IntervalList`, or `IntervalMatrix` object, respectively.

Details

Implementation of basic interval arithmetic calculations (see, for example, Moore *et al*., 2009) through Minkowski's sum (see Minkowski, 1903) and a product by a scalar operation. In particular, `+`, `-`, and `*` operators allow to carry out these computations. Using *mid/spr*-characterization, these operations can be settled for any two interval-valued data A and B and a real number γ as follows:

$$A + B = [(\text{mid } A + \text{mid } B) \mp (\text{spr } A + \text{spr } B)]$$

and

$$\gamma \cdot A = \begin{cases} [\gamma \cdot \text{mid } A \mp \gamma \cdot \text{spr } A] & \text{if } \gamma \geq 0, \\ [\gamma \cdot \text{mid } A \pm \gamma \cdot \text{spr } A] & \text{if } \gamma < 0. \end{cases}$$

Value

This function returns a single interval or a list or matrix of intervals, that is, an `IntervalData` object or an `IntervalList` or `IntervalMatrix` instance, containing the result of the involved operation. When these binary operators are used with `IntervalList` and `IntervalMatrix` objects, they return the result of the element by element operations recycling the elements of the object with the shortest dimensions if necessary, showing a warning when they are recycled only fractionally.

Author(s)

José García-García <garciagarjose@uniovi.es>

References

- Minkowski, H. (1903). Volumen und oberfläche. *Mathematische Annalen*, 57:447-495.
- Moore, R.E.; Kearfott, R.B.; Cloud, M.J. (2009). Introduction to Interval Analysis. *Society for Industrial and Applied Mathematics*, USA. doi:10.1137/1.9780898717716.

Examples

```
## Some basic arithmetic interval operations

## IntervalData
i1 <- IntervalData(0, 1)
i2 <- IntervalData(2, 3)

i1 + i2   ## Sum of two intervals
i1 + 1    ## Sum of an interval and a real number
```

```
1 + i1    ## Sum of a real number and an interval

i1 - i2   ## Subtraction of two intervals
i1 - i1   ## Note that i1 - i1 is not {0}
i1 - 1    ## Subtraction of an interval and a real number
1 - i1    ## Subtraction of a real number and an interval
- i1

2 * i1    ## Product between a scalar and an interval
-2 * i1   ## Product between a scalar and an interval
i1 * 2    ## Product between an interval and a scalar
i1 * (-2) ## Product between an interval and a scalar

## IntervalList
list1 <- IntervalList(c(0, 3, 2, 5), c(4, 5, 4, 8))
list2 <- IntervalList(c(3, 0, 3, 1), c(7, 4, 6, 2))

list1 + list2   ## Sum of two list of intervals
list1 + 1       ## Sum of a list of intervals and a real number
1 + list1       ## Sum of a real number and a list of intervals
1:4 + list1     ## Sum of a vector and a list of intervals
list1 + 1:4     ## Sum of a list of intervals and a vector

list1 - list2   ## Subtraction of two lists of intervals
list1 - 1       ## Subtraction of a list of intervals and a real number
1 - list1       ## Subtraction of a real number and a list of intervals
1:4 - list1     ## Subtraction of a vector and a list of intervals
list1 - 1:4     ## Subtraction of a list of intervals and a vector
- list1

2 * list1       ## Product between a scalar and a list of intervals
-2 * list1      ## Product between a scalar and a list of intervals
list1 * 2       ## Product between a list of intervals and a scalar
list1 * (-2)    ## Product between a list of intervals and a scalar
1:4 * list1     ## Product between a vector and a list of intervals
list1 * 1:4     ## Product between a list of intervals and vector

## IntervalMatrix
matrix1 <- IntervalMatrix(matrix(c(0, 1, 1, 2, 2, 3, 3, 4), 2, 4))
matrix2 <- IntervalMatrix(matrix(c(4, 5, 5, 6, 6, 7, 7, 8), 2, 4))
m <- matrix(1:4, 2, 2)

matrix1 + matrix2  ## Sum of two matrices of intervals
matrix1 + 1        ## Sum of a matrix of intervals and a scalar
1 + matrix1        ## sum of a scalar and a matrix of intervals
matrix1 + m        ## Sum of a matrix of intervals and a matrix
m + matrix1        ## Sum of a matrix and a matrix of intervals
matrix1 + i1       ## Sum of a matrix of intervals and an interval
i1 + matrix1       ## Sum of an interval and a matrix of intervals
matrix1 + list1    ## Sum of a matrix and a list of intervals
list1 + matrix1    ## Sum of a list and a matrix of intervals

matrix1 - matrix2  ## Subtraction of two matrices of intervals
```

```
matrix1 - 1        ## Subtraction of a matrix of intervals and a scalar
1 - matrix1        ## Subtraction of a scalar and a matrix of intervals
matrix1 - m        ## Subtraction of a matrix of intervals and a matrix
m - matrix1        ## Subtraction of a matrix and a matrix of intervals
matrix1 - i1       ## Subtraction of a matrix of intervals and an interval
i1 - matrix1       ## Subtraction of an interval and a matrix of intervals
matrix1 - list1    ## Subtraction of a matrix and a list of intervals
list1 - matrix1    ## Subtraction of a list and a matrix of intervals
- matrix1

matrix1 * 2        ## Product between a matrix of intervals and a scalar
2 *  matrix1       ## Product between a scalar and a matrix of intervals
matrix1 * m        ## Product between a matrix of intervals and a matrix
m * matrix1        ## Product between a matrix and a matrix of intervals
```

`as.IntervalData` *Convert a real number into a degenerate interval*

Description

This function allows to coerce a real number stored as a single `numeric` object to a degenerate interval formed only by this real number saved as an `IntervalData` instance.

Usage

```
## S4 method for signature 'numeric'
as.IntervalData(object)
```

Arguments

`object`	A single real number stored as a single `numeric` object.

Details

Single real numbers could be seen as particular cases of interval-valued data where each interval's lower and upper bounds are equal or, alternatively, its spread is zero.

Value

This function returns a degenerate interval saved as an object of class `IntervalData`.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Other coercion function is `as.IntervalList()`.

Examples

```
## Transform a single real-valued number into an interval
## In particular, degenerate interval {1} is defined.
i <- as.IntervalData(1); i
```

`as.IntervalList` *Convert a single interval to a list of intervals*

Description

This function allows to coerce a single interval saved as an `IntervalData` object to a degenerated list of intervals composed only of this interval and stored as an `IntervalList` instance.

Usage

```
## S4 method for signature 'IntervalData'
as.IntervalList(object)
```

Arguments

`object`	A single interval stored as an `IntervalData` object.

Value

This function returns the interval coerced to a list of intervals stored as an `IntervalList` object.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Other coercion function is `as.IntervalData()`.

Examples

```
## Convert a single interval into a list of intervals with a single interval
## In particular, degenerate interval list {[0, 1]} = {[1 -+ 1]} is defined.
as.IntervalList(IntervalData(0, 1))
```

c *Combine* IntervalData *and* IntervalList *objects*

Description

This function allows to combine single intervals and lists of intervals, that is, IntervalData and IntervalList objects, and then store the attained result as an IntervalList instance.

Usage

```
## S4 method for signature 'IntervalDataOrIntervalList'
c(x, ...)
```

Arguments

x	A single nonempty compact real interval or a unique list of of this family stored as an IntervalData object or an IntervalList instance, respectively.
...	Additional single nonempty compact real intervals or lists of intervals of this family stored as IntervalData or IntervalList instances, respectively.

Value

This function returns the list of intervals obtained after the combination of the given interval-valued elements saved as anIntervalList object.

Author(s)

José García-García <garciagarjose@uniovi.es>

Examples

```
## Combine 'IntervalData' objects
i1 <- IntervalData(0, 1)
i2 <- IntervalData(0, 2)
list1 <- c(i1, i2)
list1

## Combine 'IntervalList' objects
list2 <- c(list1, list1)
list2

## Combine both 'IntervalData' and 'IntervalList' objects
list3 <- c(i1, list1)
list3
list4 <- c(list1, i1)
list4
```

cbind *Combine* IntervalList *and* IntervalMatrix *objects by columns*

Description

This function allows to combine a sequence of lists and matrices of intervals, that is, IntervalList and IntervalMatrix objects, by columns, and store the result as an IntervalMatrix instance.

Usage

```
## S4 method for signature 'IntervalListOrIntervalMatrix'
cbind(..., deparse.level = 1)
```

Arguments

...	A sequence of lists or matrices of nonempty compact real intervals stored as IntervalList or IntervalMatrix objects, respectively.
deparse.level	Currently not used (put here to match the signature of the base implementation).

Value

This function returns a matrix of interval-valued data stored as an IntervalMatrix object.

Author(s)

José García-García <garciagarjose@uniovi.es>

Examples

```
## Some cbind() examples
list1 <- IntervalList(c(0, 3), c(4, 5))
list2 <- IntervalList(c(3, 0), c(7, 4))
cbind(list1, list2)

matrix1 <- IntervalMatrix(matrix(c(0, 1, 2, 3, 0, 3, 4, 9), 2, 4))
matrix2 <- IntervalMatrix(matrix(c(1, 5, 2, 6, 0, 1, 2, 3), 2, 4))
cbind(matrix1, matrix2)

cbind(list1, matrix1)
```

cov *Calculate the sample covariance between two random intervals*

Description

This function calculates the sample covariance between two realizations of n nonempty compact real intervals drawn from two random intervals saved as two different `IntervalList` objects.

Usage

```
## S4 method for signature 'IntervalList,IntervalList'
cov(x, y, theta = 1)
```

Arguments

`x`	A list of intervals, that is, an `IntervalList` object.
`y`	A list of intervals, that is, an `IntervalList` object with the same length as `x`.
`theta`	A single positive real number saved as a `numeric` object. By default, `theta = 1`.

Details

Let $\mathcal{X}$ and $\mathcal{Y}$ be two interval-valued random sets and let $((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$ be a sample of n independent observations drawn from $(\mathcal{X}, \mathcal{Y})$. Then, the sample covariance between $\mathcal{X}$ and $\mathcal{Y}$ is defined as the following real number given by

$$s_{\mathcal{X}\,\mathcal{Y}} = s_{\mathrm{mid}\,\mathcal{X}\,\mathrm{mid}\,\mathcal{Y}} + \theta \cdot s_{\mathrm{spr}\,\mathcal{X}\,\mathrm{spr}\,\mathcal{Y}},$$

where $\theta > 0$ and

$$s_{\mathrm{mid}\,\mathcal{X}\,\mathrm{mid}\,\mathcal{Y}} = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{mid}\,x_i - \mathrm{mid}\,\overline{x})(\mathrm{mid}\,y_i - \mathrm{mid}\,\overline{y}),$$

$$s_{\mathrm{spr}\,\mathcal{X}\,\mathrm{spr}\,\mathcal{Y}} = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{spr}\,x_i - \mathrm{spr}\,\overline{x})(\mathrm{spr}\,y_i - \mathrm{spr}\,\overline{y}),$$

with $\overline{x}$ and $\overline{y}$ being the sample Aumann means of the given one-dimensional random samples.

Value

This function returns the calculated sample covariance of two samples of n interval-valued data, which is defined as a real number. Therefore, the output of this function is a single `numeric` value.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Other sample central tendency and dispersion measures such as sample Aumann mean and sample Fréchet variance can be calculated through `mean()` and `var()` functions, respectively.

Examples

```
## Some cov() examples changing theta
list1 <- IntervalList(c(0, 3, 2, 5, 6), c(4, 5, 4, 8, 7))
list2 <- IntervalList(c(3, 0, 3, 1, 4), c(7, 4, 6, 2, 6))
cov(list1, list2)
cov(list1, list2, 1/3)

## Note that cov(X, X) = var(X)
cov(list1, list1)
var(list1)
cov(list1, list1, 1/3)
var(list1, 1/3)
```

---

cronbach *Estimate sample Cronbach's α coefficient*

Description

This function allows to calculate the sample Cronbach's α coefficient as an estimate of the reliability (understood in this case under the internal consistency point of view) of the responses collected through Likert-type, visual analogue, and interval-valued rating scales in questionnaires.

Usage

```
cronbach(data, ivs = TRUE, type = 1, theta = 1)
```

Arguments

`data`	A `matrix` or `data.frame` with the questionnaire's responses.
`ivs`	A `logical` value indicating if an interval-valued scale is used (default) or not.
`type`	A single `numeric` value specifying both the order and the characterization that is being used for storing interval-valued information on `data` argument. Only the following four options are allowed: • `1`: The *inf/sup*-characterization is used variable by variable (default). • `2`: The *mid/spr*-characterization is used variable by variable. • `3`: All the supremums follow all the infimums in the same variable order. • `4`: All the spreads follow all the mid-points in the same variable order.
`theta`	A single positive real number stored as a unique `numeric` value which is used for distance computations. By default, `theta = 1`.

Details

For both traditional Likert-type and visual analogue rating scales responses, the sample Cronbach's α coefficient (Cronbach, 1951) computed by `cronbach()` function is defined as follows,

$$\widehat{\alpha} = \frac{k}{k-1}\left(1 - \frac{\sum_{j=1}^{k} s^2_{X_j}}{s^2_{X_{total}}}\right),$$

where $k > 1$ is the number of items; $s^2_{X_j}$ is the sample variance of X_j, which is the real-valued random variable modeling the responses to the j-th item; and $s^2_{X_{total}}$ is the sample variance of the sum of all the involved items, that is,

$$X_{total} = X_1 + X_2 + \ldots + X_k.$$

Analogously, for interval-valued scale responses the sample Cronbach's α coefficient computed by this function is defined as follows,

$$\widehat{\alpha} = \frac{k}{k-1}\left(1 - \frac{\sum_{j=1}^{k} s^2_{\mathcal{X}_j}}{s^2_{\mathcal{X}_{total}}}\right),$$

where $k > 1$ is the number of items; $s^2_{\mathcal{X}_j}$ is the sample Fréchet variance of $\mathcal{X}_j$, which is the interval-valued random set modeling the responses to the j-th item; and $s^2_{\mathcal{X}_{total}}$ is the sample Fréchet variance of the sum of all the involved items, that is,

$$\mathcal{X}_{total} = \mathcal{X}_1 + \mathcal{X}_2 + \ldots + \mathcal{X}_k.$$

Value

This function returns the calculated sample Cronbach's α coefficient stored as a single `numeric` value for measuring the reliability or internal consistency of the given questionnaire's responses.

Author(s)

José García-García <garciagarjose@uniovi.es>

References

Cronbach L.J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16, 297-334. doi:1001007/BF02310555.

Examples

```
## These code lines illustrate Cronbach's alpha coefficient calculation
## for interval-valued, Likert-type, and visual analogue scales responses

## Some trivial cronbach() examples
## Cronbach's alpha index for interval-valued scale responses stored
## in a matrix with the inf/sup-characterization variable by variable
## using Bertoluzza's distance with Lebesgue measure (theta = 1/3)
data1 <- matrix(c(1, 2.6, 1.5, 3, 3.8, 6, 4, 7), 2, 4)
```

```
cronbach(data1, theta = 1/3)

## Cronbach's alpha index for interval-valued scale responses stored
## in a data.frame with the mid/spr-characterization saving all the
## mid-points and then all the spreads using rho2 distance (theta = 1)
data2 <- data.frame(mids1 = c(2, 3),
                    mids2 = c(4, 5),
                    sprs1 = c(1, 2),
                    sprs2 = c(2, 4))
cronbach(data2, type = 4)

## Cronbach's alpha coefficient for Likert-type
## scale responses stored in a matrix
data3 <- matrix(c(1, 3, 4, 7), 2, 2)
cronbach(data3, ivs = FALSE)

## Cronbach's alpha coefficient for visual analogue
## scale responses stored in a data.frame
data4 <- data.frame(item1 = c(1.5, 2.8),
                    item2 = c(3.9, 6.2))
cronbach(data4, ivs = FALSE)

## Real-life data example
## Load the interval-valued data
data(lackinfo, package = "IntervalQuestionStat")

## Calculate Cronbach's alpha coefficient for interval-valued responses
cronbach(lackinfo[, 3:12])

## Convert interval-valued responses into their corresponding equivalent
## Likert-type answers and then calculate Cronbach's alpha coefficient
likert <- ivs2likert(IntervalMatrix(lackinfo[, 3:12]))
cronbach(likert, ivs = FALSE)

## Analogously, interval-valued responses are transformed into their
## corresponding equivalent visual analogue scale answers and
## Cronbach's alpha coefficient is then computed
vas <- ivs2vas(IntervalMatrix(lackinfo[, 3:12]))
cronbach(vas, ivs = FALSE)
```

`dim` *Get the number of rows and columns of an* `IntervalMatrix` *object*

Description

This function allows to get the number of rows and columns of an interval-valued matrix.

Usage

```
## S4 method for signature 'IntervalMatrix'
dim(x)
```

Arguments

x	A matrix of interval-valued data stored as an `IntervalMatrix` object.

Value

This function returns a bidimensional vector with the number of rows and columns, respectively, of an interval-valued matrix. Therefore, it always returns an `integer` object whose length is two.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

The **IntervalQuestionStat** package also allows to obtain the number of rows and columns of an `IntervalMatrix` object separately through `nrow()` and `ncol()` functions, respectively.

Examples

```
## Some dim() examples

data1 <- matrix(c(0, 1, 2, 3, 0, 3, 4, 9), 2, 4)
m1 <- IntervalMatrix(data1)
dim(m1)

data2 <- matrix(c(1, 5, 3, 2, 6, 4, 0, 1, 3,
                  2, 3, 9, 4, 3, 7, 5, 6, 8), 3, 6)
m2 <- IntervalMatrix(data2)
dim(m2)
```

---

distance *Calculate the θ-distance between two intervals*

Description

This function calculates the θ-distance between any two nonempty compact real intervals.

Usage

```
## S4 method for signature 'IntervalData,IntervalData'
distance(e1, e2, theta = 1)
```

Arguments

e1	A single interval stored as an IntervalData object.
e2	A single interval stored as an IntervalData object.
theta	A single positive real number stored as a unique numeric value which is used for distance computations. By default, theta = 1.

Details

The θ-distance between any two given nonempty compact real intervals, A and B, was defined by Gil *et al*. (2002) as the non-negative real number calculated as follows,

$$d_\theta(A, B) = \sqrt{(\text{mid } A - \text{mid } B)^2 + \theta \cdot (\text{spr } A - \text{spr } B)^2},$$

where θ is a positive real number.

Value

This function returns the calculated θ-distance between the two given intervals, which is defined as a single real number. Therefore, the output of this function is a single numeric value.

Author(s)

José García-García <garciagarjose@uniovi.es>

References

Gil, M.Á.; Lubiano, M.A.; Montenegro, M.; López, M.T. (2002). Least squares fitting of an affine function and strength of association for interval-valued data. *Metrika*, 56:97-111. doi:10.1007/s001840100160.

Examples

```
## Some distance() examples
i1 <- IntervalData(0, 1)
i2 <- IntervalData(3, 7)
distance(i1, i2)      ## rho2 distance
distance(i1, i2, 1/3) ## Bertoluzza's distance with Lebesgue measure
```

---

extract *Extract and replace parts of an interval-valued object*

Description

This command allows to extract and replace parts of interval-valued lists or matrices.

Usage

```
## S4 method for signature 'IntervalList'
x[i]

## S4 method for signature 'IntervalList'
x[[i]]

## S4 method for signature 'IntervalMatrix'
x[i, j]

## S4 replacement method for signature 'IntervalList'
x[i] <- value

## S4 replacement method for signature 'IntervalList'
x[[i]] <- value

## S4 replacement method for signature 'IntervalMatrix'
x[i, j] <- value
```

Arguments

x	A list with several nonempty compact intervals or a matrix of intervals of this family, that is, an IntervalList object or an IntervalMatrix instance.
i	The indices of the elements of the list or the rows of the elements of the matrix wanted to be extracted saved as a numeric, integer, or logical object.
value	A single nonempty compact interval or a list or matrix of intervals saved of this family stored as an IntervalData, IntervalList, or IntervalMatrix object.
j	The indices of the columns of the matrix's elements wanted to be selected and extracted saved as a numeric, integer, or logical object.

Details

Both i and j can also be negative integers, indicating the elements lo leave out of the selection.

Value

It should be remarked that, on the one hand, the [command returns the selected elements as a list of intervals, that is, an IntervalList object, when it is used for IntervalList instances. On the other hand, it returns IntervalData, IntervalList, or IntervalMatrix objects when it is used with IntervalMatrix instances. The [[command allows to extract a single interval, that is, an IntervalData object, from a list of several intervals saved as an IntervalList object. Finally, both [<- and [[<- commands allow to replace the selected elements of the given interval-valued object by another object of the required class and, thus, they do not return any value.

Author(s)

José García-García <garciagarjose@uniovi.es>

Examples

```
## Extract parts of a list of intervals
list <- IntervalList(c(1, 3, 5), c(2, 4, 6))
list[1]        ## Extract the first interval
               ## Note that the output is an IntervalList object
list[[1]]      ## The first interval is extracted as an IntervalData object
list[c(1, 3)]  ## Extract the first and the third interval

## Extract parts of a matrix of intervals
m <- IntervalMatrix(matrix(c(1, 5, 2, 6, 6, 2, 7, 3, 3, 4, 4, 5), 2, 6))
m[1, 1]        ## Extract the interval from the first row and first column
m[1,]          ## Extract all the intervals from the first row
m[, 1]         ## Extract all the intervals from the first column
m[, c(1, 3)]   ## Extract the sub-matrix containing all the intervals
               ## from both first and third column of the original matrix

## Replace parts of a list of intervals
list[[1]]
list[[1]] <- IntervalData(0, 1)
list[[1]]

list[c(1, 3)]
list[c(1, 3)] <- IntervalList(c(0, 1), c(1, 2))
list[c(1, 3)]

## Replace parts of a matrix of intervals
m[1, 1]
m[1, 1] <- IntervalData(0, 1)
m[1, 1]

m[1, 1:2]
m[1, 1:2] <- IntervalList(c(0, 1), c(1, 2))
m[1, 1:2]

m[, c(1, 3)]
m[, c(1, 3)] <- IntervalMatrix(matrix(1:8, 2, 4))
m[, c(1, 3)]
```

`IntervalData` *Create an* `IntervalData` *object*

Description

For convenience, `IntervalData` objects or instances may be created with this function.

Usage

```
IntervalData(a1, a2, type = 1)
```

Arguments

a1	A single real number specifying either the infimum or either the mid-point of the interval stored as a unique numeric value.
a2	A single real number specifying either the supremum or either the spread of the interval stored as a unique numeric value.
type	A single real number specifying the characterization that is being used stored as a unique numeric value. Only two options are allowed: • 1: The *inf/sup*-characterization is used (default). • 2: The *mid/spr*-characterization is used.

Value

This function returns the created IntervalData object.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

For other interval-valued data definition use IntervalList() and IntervalMatrix() functions.

Examples

```
## The following code generates the same interval through
## both inf/sup and mid/spr characterizations, respectively.
## In particular, interval [0, 2] = [1 -+ 1] is defined.
i1 <- IntervalData(a1 = 0, a2 = 2, type = 1); i1
i2 <- IntervalData(a1 = 1, a2 = 1, type = 2); i2
```

IntervalData-class *S4 class representing a single interval*

Description

Each nonempty compact real interval K can be alternatively characterized in terms of either its lower and upper bounds (also called infimum and supremum, respectively) through what is usually known as its *inf/sup*-characterization or either its mid-point and spread (also named center and radius, respectively) by means of its *mid/spr*-characterization as follows,

$$K = [\inf K, \sup K] = [\operatorname{mid} K \mp \operatorname{spr} K],$$

where both $\inf K \leq \sup K$ and $\operatorname{spr} K \geq 0$ conditions are fulfilled. The existing equivalence relation between these two characterizations is given by the following two equations:

$$\operatorname{mid} K = \frac{\sup K + \inf K}{2} \qquad \text{and} \qquad \operatorname{spr} K = \frac{\sup K - \inf K}{2}.$$

Slots

`mid`: A single real number saved as a unique `numeric` value specifying the mid-point of the interval.

`spr`: A single real number saved as a unique `numeric` value specifying the spread of the interval.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Objects of `IntervalData` class should be created through `IntervalData()` function. Besides `IntervalData` class, the **IntervalQuestionStat** package also incorporates `IntervalList-class` and `IntervalMatrix-class` for dealing with interval-valued data in R environment.

Examples

```
showClass("IntervalData")
showMethods(classes = "IntervalData")
```

IntervalDataOrIntervalList-class

Virtual union of `IntervalData` *and* `IntervalList` *classes*

Description

This virtual class is defined as a superclass of `IntervalData` and `IntervalList` classes.

Details

In particular, this virtual class is defined only for internal use in the **IntervalQuestionStat** package implementation since it is useful for supplying several classes in some methods signatures. It is very important to remark that users can not define instances of this virtual class.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Besides `IntervalDataOrIntervalList`, other virtual class defined only for internal purposes in the **IntervalQuestionStat** package is `IntervalListOrIntervalMatrix-class`.

Examples

```
showClass("IntervalDataOrIntervalList")
showMethods(classes = "IntervalDataOrIntervalList")
```

`IntervalList` *Create an* `IntervalList` *object*

Description

For convenience, `IntervalList` objects or instances may be created with this function.

Usage

```
IntervalList(x, y = NULL, type = 1)
```

Arguments

`x`	A `numeric` vector, `matrix` or `data.frame`.
`y`	`NULL` (default) or a `numeric` vector with compatible dimensions to `x`.
`type`	A single real number specifying the characterization that is being used stored as a unique `numeric` value. Only the following two options are allowed: • `1`: The *inf/sup*-characterization is used (default). • `2`: The *mid/spr*-characterization is used.

Details

In order to create an `IntervalList` object, the information that defines the intervals of the list (either the lower and upper bounds or either the mid-points and the spreads) can be given as input to `IntervalList()` function in two different ways. On the one hand, each type of characterizing point can be stored separately in two `numeric` vectors and then they are passed to the function through `x` and `y` arguments. On the other hand, they can be stored jointly in a `matrix` or `data.frame` and then only `x` argument is used (`y` is left as `NULL` as it is defined by default).

Value

This function returns the created `IntervalList` object.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

For other interval-valued data definition use `IntervalData()` and `IntervalMatrix()` functions.

Examples

```
## The following code generates the same list of intervals in four
## different ways by using different characterizations. In particular,
## the following list made up of three intervals is defined,
## {[0, 1], [2, 6], [5, 10] = [0.5 -+ 0.5], [4 -+ 2], [7.5 -+ 2.5]}.

## First, inf/sup-characterization stored in two vectors is used.
list1 <- IntervalList(c(0, 2, 5), c(1, 6, 10)); list1
## Then, mid/spr-characterization stored in two vectors is used.
list2 <- IntervalList(c(0.5, 4, 7.5), c(0.5, 2, 2.5), type = 2); list2
## Then, inf/sup-characterization stored in a matrix is used.
matrix <- matrix(c(0, 2, 5, 1, 6, 10), 3, 2)
list3 <- IntervalList(matrix); list3
## Finally, mid/spr-characterization stored in a data.frame is used.
dataframe <- data.frame(mids = c(0.5, 4, 7.5), sprs = c(0.5, 2, 2.5))
list4 <- IntervalList(dataframe, type = 2); list4
```

IntervalList-class *S4 class representing a list of intervals*

Description

S4 class representing a list of intervals

Slots

`mid`: A vector of real numbers saved as a `numeric` object specifying the mid-points of the intervals.

`spr`: A vector of real numbers saved as a `numeric` object specifying the spreads of the intervals.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Objects of `IntervalList` class should be created through `IntervalList()` function. Besides `IntervalList` class, the **IntervalQuestionStat** package also incorporates `IntervalData-class` and `IntervalMatrix-class` for dealing with interval-valued data in R environment.

Examples

```
showClass("IntervalList")
showMethods(classes = "IntervalList")
```

IntervalListOrIntervalMatrix-class *Virtual union of* IntervalList *and* IntervalMatrix *classes*

Description

This virtual class is defined as a superclass of IntervalList and IntervalMatrix classes.

Details

In particular, this virtual class is defined only for internal use in the **IntervalQuestionStat** package implementation since it is useful for supplying several classes in some methods signatures. It is very important to remark that users can not define instances of this virtual class.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Besides IntervalListOrIntervalMatrix, other virtual class defined only for internal purposes in the **IntervalQuestionStat** package is IntervalDataOrIntervalList-class.

Examples

```
showClass("IntervalListOrIntervalMatrix")
showMethods(classes = "IntervalListOrIntervalMatrix")
```

IntervalMatrix *Create an* IntervalMatrix *object*

Description

For convenience, IntervalMatrix objects or instances may be created with this function.

Usage

```
IntervalMatrix(data, type = 1)
```

Arguments

data	A matrix or data.frame with the interval-valued information.
type	A single numeric value specifying both the order and the characterization that is being used for storing interval-valued information on data argument. Only the following four options are allowed: • 1: The *inf/sup*-characterization is used variable by variable (default). • 2: The *mid/spr*-characterization is used variable by variable. • 3: All the supremums follow all the infimums in the same variable order. • 4: All the spreads follow all the mid-points in the same variable order.

Value

This function returns the created IntervalMatrix object.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

For other interval-valued data definition use IntervalData() and IntervalList() functions.

Examples

```
## The following code illustrates four different ways to define
## the same matrix of intervals through IntervalMatrix() function.
## In particular, the following 2x2 interval-valued matrix is defined:
## | [0, 2] [0, 4] | = | [1 -+ 1] [2 -+ 2] |
## | [1, 3] [3, 9] | = | [2 -+ 1] [6 -+ 3] |

## Using inf/sup characterization variable by variable
data1 <- matrix(c(0, 1, 2, 3, 0, 3, 4, 9), 2, 4)
matrix1 <- IntervalMatrix(data1); matrix1
## Using mid/spr characterization variable by variable
data2 <- matrix(c(1, 2, 1, 1, 2, 6, 2, 3), 2, 4)
matrix2 <- IntervalMatrix(data2, type = 2); matrix2
## Storing all the infimums first and then all the supremums
data3 <- matrix(c(0, 1, 0, 3, 2, 3, 4, 9), 2, 4)
matrix3 <- IntervalMatrix(data3, type = 3); matrix3
## Storing all the mid-points first and then all the spreads
data4 <- matrix(c(1, 2, 2, 6, 1, 1, 2, 3), 2, 4)
matrix4 <- IntervalMatrix(data4, type = 4); matrix4
```

IntervalMatrix-class *S4 class representing a matrix of intervals*

Description

S4 class representing a matrix of intervals

Slots

`mid`: A matrix of real numbers saved as a `matrix` object specifying the mid-points of the intervals.

`spr`: A matrix of real numbers saved as a `matrix` object specifying the spreads of the intervals.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Objects of `IntervalMatrix` class should be created through `IntervalMatrix()` function. Besides `IntervalMatrix` class, the **IntervalQuestionStat** package also incorporates `IntervalData-class` and `IntervalList-class` for dealing with interval-valued data in R environment.

Examples

```
showClass("IntervalMatrix")
showMethods(classes = "IntervalMatrix")
```

ivs2likert *Convert interval-valued responses into their equivalent numerically encoded Likert-type scale answers with minimum θ-distance criterion*

Description

This function allows to associate each nonempty compact real interval collected as a response in a questionnaire designed with interval-valued scales to its equivalent numerically encoded Likert-type scale answer following the minimum θ-distance criterion.

Usage

```
ivs2likert(x, k = 7, minimum = 1, maximum = 7, theta = 1)
```

Arguments

x	Either a single interval or either a list or matrix with several intervals stored as an IntervalData object or as an IntervalList or IntervalMatrix instance.
k	A single positive integer number stored as a numeric object which indicates the number of different Likert-type responses to be considered. By default, k = 7.
minimum	A single real number indicating the lower bound of the interval-valued scale used saved as a unique numeric value. By default, minimum = 1.
maximum	A single real number indicating the upper bound of the interval-valued scale used saved as a unique numeric value. By default, maximum = 7.
theta	A single positive real number stored as a unique numeric value which is used for distance computations. By default, theta = 1.

Details

If a k-point Likert-type scale with reference interval $[l, u]$ is considered, then the minimum distance criterion consists on associating each interval-valued scale response with the real number in the set defined by $\{L_1, L_2, \ldots, L_k\}$, where each L_i is defined as follows,

$$L_i = l + (i-1)\frac{u-l}{k-1}, \qquad i = 1, 2, \ldots, k,$$

with the smallest θ-distance to the given data. That is, each interval A is associated with the real number $L(A)$ such that

$$L(A) = \arg \min_{L \in \{L_1, L_2, \ldots, L_k\}} d_\theta\left(A, \{L\}\right).$$

If ties are produced, they are broken at random.

Value

This function returns the nearest Likert-type responses for the given interval-valued data following the minimum θ-distance criterion stored either as a numeric object if x argument is a single interval or a list of intervals, that is, an IntervalData or IntervalList instance, or either as a data.frame object whether x is a matrix of intervals, that is, an IntervalMatrix object.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Interval-valued responses can be also associated to their corresponding answers in a visual analogue scale through the mid-point criterion implemented in the ivs2vas() function.

Examples

```
## Some ivs2likert() examples using an interval-valued scale bounded
## between 0 and 10, a 11-point Likert scale, and rho2 distance (theta = 1)

## A single interval-valued response
```

```
i <- IntervalData(3, 3.2)
ivs2likert(i, k = 11, minimum = 0, maximum = 10)

## A list of interval-valued responses
list <- IntervalList(c(3, 8.7), c(3.2, 9))
ivs2likert(list, k = 11, minimum = 0, maximum = 10)

## A matrix of interval-valued responses
matrix <- IntervalMatrix(matrix(c(1, 2.6, 1.5, 3, 3.8, 6, 4, 7), 2, 4))
ivs2likert(matrix, k = 11, minimum = 0, maximum = 10)
```

`ivs2vas` *Convert interval-valued responses into equivalent visual analogue scale answers through mid-point criterion*

Description

This function allows to reduce each nonempty compact real interval collected as a response in a questionnaire designed with interval-valued scales to its mid-point so it can be considered as an answer from a visual analogue scale. That is, given an nonempty compact real, interval, A, this method returns its mid-point, which can be calculated as follows,

$$\operatorname{mid} A = \frac{\sup + \inf A}{2}.$$

Usage

```
ivs2vas(x)
```

Arguments

`x`	Either a single interval or either a list or matrix with several intervals stored as an `IntervalData` object or as an `IntervalList` or `IntervalMatrix` instance.

Value

This function returns the mid-points of the given interval-valued data to be considered as visual analogue scale responses in a questionnaire stored either as a `numeric` object if `x` argument is a single interval or a list of intervals, that is, an `IntervalData` or `IntervalList` instance, or either as a `data.frame` object whether `x` is a matrix of intervals, that is, an `IntervalMatrix` object.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

Interval-valued responses can be also associated to their corresponding Likert-type scale's answer through the minimum θ-distance criterion implemented in the `ivs2likert()` function.

Examples

```
## Some ivs2vas() examples

## A single interval-valued response
i <- IntervalData(3, 3.2)
ivs2vas(i)

## A list of interval-valued responses
list <- IntervalList(c(3, 8.7), c(3.2, 9))
ivs2vas(list)

## A matrix of interval-valued responses
matrix <- IntervalMatrix(matrix(c(1, 2.6, 1.5, 3, 3.8, 6, 4, 7), 2, 4))
ivs2vas(matrix)
```

---

lackinfo *Lack of information in expository face-to-face lessons data set*

Description

A data set containing the interval-valued responses to 5 different items measuring the perception of lack of information in expository face-to-face lessons obtained with a questionnaire carried out at the University of Oviedo (Spain). Respondents sex is also included.

Usage

```
lackinfo
```

Format

A data frame with 50 observations of the following 12 variables:

- `id`: identification number.
- `sex`: sex of the respondent (`male` or `female`).
- `inf1`: infimum of respondent's interval-valued answer to item 1.
- `sup1`: supremum of respondent's interval-valued answer to item 1.
- `inf2`: infimum of respondent's interval-valued answer to item 2.
- `sup2`: supremum of respondent's interval-valued answer to item 2.
- `inf3`: infimum of respondent's interval-valued answer to item 3.
- `sup3`: supremum of respondent's interval-valued answer to item 3.
- `inf4`: infimum of respondent's interval-valued answer to item 4.
- `sup4`: supremum of respondent's interval-valued answer to item 4.
- `inf5`: infimum of respondent's interval-valued answer to item 5.
- `sup5`: supremum of respondent's interval-valued answer to item 5.

Details

An educational innovation project was carried out for improving teaching-learning processes at the University of Oviedo (Spain) for the 2020/2021 academic year. A total of 50 students have been requested to answer an online questionnaire about their perception of lack of information in expository face-to-face lessons by selecting the interval that best represents their level of agreement to the statements proposed in a interval-valued scale bounded between 1 and 7, where 1 means *strongly disagree* and 7 represents the option *strongly agree*.

These are the 5 items used to measure the perception of the students about the lack of information in expository face-to-face lessons:

1. I receive too little information from my classmates.
2. It is difficult to receive relevant information from my classmates.
3. It is difficult to receive relevant information from the teacher.
4. The amount of information I receive from my classmates is very low.
5. The amount of information I receive from the teacher is very low.

Moreover, the students have been requested to indicate their sex too.

Examples

```
## Real-life data set with interval-valued responses to a questionnaire

## Load the data set
data(lackinfo, package = "IntervalQuestionStat")

## Explore the data set: first rows, structure, and summary
head(lackinfo)
str(lackinfo)
summary(lackinfo)
```

`length` *Get the length of an* `IntervalList` *object*

Description

This function allows to get the length of an interval-valued list, that is, the number of different nonempty compact real intervals stored in an `IntervalList` object.

Usage

```
## S4 method for signature 'IntervalList'
length(x)
```

Arguments

`x`	A list of nonempty compact real intervals stored as an `IntervalList` object.

Value

This function returns a single numeric value indicating the length of a list with several nonempty compact real intervals. Therefore, it always returns an `integer` object of whose length is one.

Author(s)

José García-García <garciagarjose@uniovi.es>

Examples

```
## Some length() examples

list1 <- IntervalList(c(0, 2, 5), c(1, 6, 10))
length(list1)

dataframe <- data.frame(mids = c(0.5, 4, 7.5, 5),
                        sprs = c(0.5, 2, 2.5, 1))
list2 <- IntervalList(dataframe, type = 2)
length(list2)
```

---

mean *Calculate the sample Aumann mean of a random interval*

Description

This function calculates the sample Aumann mean of a single realization formed by n nonempty compact real intervals drawn from a random interval saved as an `IntervalList` object.

Usage

```
## S4 method for signature 'IntervalList'
mean(x)
```

Arguments

`x`	A list of intervals, that is, an `IntervalList` object.

Details

Let $\mathcal{X}$ be an interval-valued random set and let $(x_1, x_2, \ldots, x_n)$ be a sample of n independent observations drawn from $\mathcal{X}$. Then, the sample Aumann mean (see Aumann, 1965) is defined as the following interval given by

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

Value

This function returns an `IntervalData` object with the calculated sample Aumann mean of the given n intervals, which is defined as another nonempty compact real interval.

Author(s)

José García-García <garciagarjose@uniovi.es>

References

Aumann, R.J. (1965). Integrals of set-valued functions. *Journal of Mathematical Analysis and Applications*, 12(1):1-12. doi:10.1016/0022247X(65)900491.

See Also

Other sample dispersion and covariance measures such as sample Fréchet variance and sample covariance can be calculated through `var()` and `cov()` functions, respectively.

Examples

```
## Some mean() trivial examples
list <- IntervalList(c(1, 3), c(2, 5))
mean(list)
```

---

mid *Extract and replace the mid-points of interval-valued objects*

Description

This functions provides customized access to the `mid` slot of `IntervalData`, `IntervalList`, and `IntervalMatrix` objects, so the mid-points of the intervals can be extracted and replaced. This does not prevent to use the @ accessor, but does not force others to know the implementation details.

Usage

```
## S4 method for signature 'IntervalData'
mid(object)

## S4 method for signature 'IntervalList'
mid(object)

## S4 method for signature 'IntervalMatrix'
mid(object)

## S4 replacement method for signature 'IntervalData'
mid(object) <- value

## S4 replacement method for signature 'IntervalList'
mid(object) <- value

## S4 replacement method for signature 'IntervalMatrix'
mid(object) <- value
```

Arguments

object	A single nonempty compact interval or a list or matrix with some intervals of this family, that is, an IntervalData, IntervalList, or IntervalMatrix instance.
value	A numeric or matrix object with the new values of the intervals mid-points.

Value

On the one hand, mid() function returns the mid-points of the intervals contained in IntervalData, IntervalList, or IntervalMatrix instances stored as a single numeric value, a numeric vector, or a matrix object, respectively. On the other hand, mid<- command does not return any value since it only allows to replace the mid slot of the given interval-valued object.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

For accessing spr slot of interval-valued objects, spr() function can be used.

Examples

```
## Some mid() function examples

## With IntervalData objects
interval <- IntervalData(0, 1)

mid(interval)
mid(interval) <- 0.75
mid(interval)

## With IntervalList objects
list <- IntervalList(c(1, 3, 5), c(2, 4, 6))

mid(list[c(1, 3)])
mid(list[c(1, 3)]) <- c(1, 3)
mid(list[c(1, 3)])

## With IntervalMatrix objects
m <- IntervalMatrix(matrix(c(1, 5, 2, 6, 6, 2,
                             7, 3, 3, 4, 4, 5), 2, 6))

mid(m[1, 1])
mid(m[1, 1]) <- 2
mid(m[1, 1])

mid(m[1, 1:2])
mid(m[1, 1:2]) <- c(2, 3)
mid(m[1, 1:2])

mid(m[, c(1, 3)])
```

```
mid(m[, c(1, 3)]) <- matrix(1:4, 2, 2)
mid(m[, c(1, 3)])
```

---

ncol *Get the number of columns of an* IntervalMatrix *object*

Description

This function allows to get the number of columns of an interval-valued matrix.

Usage

```
## S4 method for signature 'IntervalMatrix'
ncol(x)
```

Arguments

x	A matrix of interval-valued data stored as an IntervalMatrix object.

Value

This function returns a single numeric value indicating the number of columns of an interval-valued matrix. Therefore, this function always returns an integer object whose length is one.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

The number of columns of an IntervalMatrix object can be obtained along with the number of rows through dim() function. In an analogous way, for getting the number of rows of an IntervalMatrix object, nrow() function can be used.

Examples

```
## Some ncol() examples

data1 <- matrix(c(0, 1, 2, 3, 0, 3, 4, 9), 2, 4)
m1 <- IntervalMatrix(data1)
ncol(m1)
data2 <- matrix(c(1, 5, 3, 2, 6, 4, 0, 1, 3,
                  2, 3, 9, 4, 3, 7, 5, 6, 8), 3, 6)
m2 <- IntervalMatrix(data2)
ncol(m2)
```

nrow *Get the number of rows of an* IntervalMatrix *object*

Description

This function allows to get the number of rows of an interval-valued matrix.

Usage

```
## S4 method for signature 'IntervalMatrix'
nrow(x)
```

Arguments

x	A matrix of interval-valued data stored as an IntervalMatrix object.

Value

This function returns a single numeric value indicating the number of rows of an interval-valued matrix. Therefore, this function always returns an integer object whose length is one.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

The number of rows of an IntervalMatrix object can be obtained along with the number of columns through dim() function. In an analogous way, for getting the number of columns of an IntervalMatrix object, ncol() function can be used.

Examples

```
## Some nrow() examples

data1 <- matrix(c(0, 1, 2, 3, 0, 3, 4, 9), 2, 4)
m1 <- IntervalMatrix(data1)
nrow(m1)
data2 <- matrix(c(1, 5, 3, 2, 6, 4, 0, 1, 3,
                  2, 3, 9, 4, 3, 7, 5, 6, 8), 3, 6)
m2 <- IntervalMatrix(data2)
nrow(m2)
```

plot *Plot a single interval or a list of intervals*

Description

S4 methods for function plot. As in the generic plot S3 'graphics' method, these methods plot interval-valued data contained in both `IntervalData` and `IntervalList` objects.

Usage

```
## S4 method for signature 'IntervalData,missing'
plot(x, y,
     layout = c("vertical", "horizontal"),
     bounds = FALSE, mid = FALSE,
     ...)

## S4 method for signature 'IntervalData,IntervalData'
plot(x, y, bounds = FALSE, ...)

## S4 method for signature 'IntervalList,missing'
plot(x, y,
     layout = c("vertical", "horizontal"),
     bounds = FALSE, mid = FALSE,
     ...)

## S4 method for signature 'IntervalList,IntervalList'
plot(x, y, bounds = FALSE, ...)
```

Arguments

`x`	A single interval or a unique list with several intervals stored as an `IntervalData` object or as an `IntervalList`, instance respectively.
`y`	A single interval or a unique list with several intervals stored as an `IntervalData` object or as an `IntervalList`, instance respectively.
`layout`	The axes along which the intervals should be displayed. Only two alternatives are allowed: `vertical` (default) and `horizontal`.
`bounds`	A single `logical` value indicating whether the extremes of the given intervals should be plotted or not (default).
`mid`	A single `logical` value indicating whether the mid-points of the given intervals should be plotted or not (default).
`...`	Other graphical parameters.

Details

Note that in order to get bidimensional plots with interval-valued data, `x` and `y` arguments must be of the same class, that is, either both are `IntervalData` objects or either both are `IntervalList` instances. Moreover, in the second case both lists must have the same number of intervals.

Value

This function does not return any value. It only plots interval-valued data.

Author(s)

José García-García <garciagarjose@uniovi.es>

Examples

```
## Some trivial plot() examples for IntervalData objects
i1 <- IntervalData(0, 1)
i2 <- IntervalData(2, 3)
plot(i1)                              ## Plot only an interval vertically
plot(i1, layout = "horizontal")       ## Plot only an interval horizontally
plot(i1, bounds = TRUE, mid = TRUE)  ## Add bounds and remark mid-point
plot(i1, i2)                          ## Plot an interval on each axis
plot(i1, i2, bounds = TRUE)           ## Add bounds

## Some trivial plot() examples for IntervalList objects
list1 <- IntervalList(c(0, 3, 2, 5, 6), c(4, 5, 4, 8, 7))
list2 <- IntervalList(c(3, 0, 3, 1, 4), c(7, 4, 6, 2, 6))
plot(list1)                             ## Plot an interval list vertically
plot(list1, layout = "horizontal")      ## Plot an interval list horizontally
plot(list1, bounds = TRUE, mid = TRUE) ## Add bounds and remark mid-points
plot(list1, list2)                      ## Plot an interval list on each axis
plot(list1, list2, bounds = TRUE)       ## Add bounds

## Further plot() customizations
plot(list1, bounds = TRUE, mid = TRUE,
     main = "My one-dimensional interval-valued plot",
     col = c("blue", "red"), lwd = 2)
plot(list1, list2, bounds = TRUE,
     main = "My bidimensional interval-valued plot",
     col = "blue", lwd = 2)
```

---

rbind *Combine* IntervalList *and* IntervalMatrix *objects by rows*

Description

This function allows to combine a sequence of lists and matrices of intervals, that is, `IntervalList` and `IntervalMatrix` objects, by rows, and store the result as an `IntervalMatrix` instance.

Usage

```
## S4 method for signature 'IntervalListOrIntervalMatrix'
rbind(..., deparse.level = 1)
```

Arguments

...	A sequence of lists or matrices of nonempty compact real intervals stored as IntervalList or IntervalMatrix objects, respectively.
deparse.level	Currently not used (put here to match the signature of the base implementation).

Value

This function returns a matrix of interval-valued data stored as an IntervalMatrix object.

Author(s)

José García-García <garciagarjose@uniovi.es>

Examples

```
## Some rbind() examples
list1 <- IntervalList(c(0, 3), c(4, 5))
list2 <- IntervalList(c(3, 0), c(7, 4))
rbind(list1, list2)

matrix1 <- IntervalMatrix(matrix(c(0, 1, 2, 3, 0, 3, 4, 9), 2, 4))
matrix2 <- IntervalMatrix(matrix(c(1, 5, 2, 6, 0, 1, 2, 3), 2, 4))
rbind(matrix1, matrix2)

rbind(list1, matrix1)
```

---

show *Print basic information of interval-valued data*

Description

This function allows to print in the console basic information of interval-valued data.

Usage

```
## S4 method for signature 'IntervalData'
show(object)

## S4 method for signature 'IntervalList'
show(object)

## S4 method for signature 'IntervalMatrix'
show(object)
```

Arguments

object	A single interval, a list of intervals or a matrix with several intervals stored as an IntervalData, IntervalList, or IntervalMatrix object.

Details

For `IntervalData` and `IntervalList` objects, both *inf/sup* and *mid/spr* characterizations of the intervals are printed, and for `IntervalMatrix` instances, the number of rows and columns is shown.

Value

This function does not return any value. It only prints the interval-valued object's information.

Author(s)

José García-García <garciagarjose@uniovi.es>

Examples

```
## Show an interval-valued data
i <- IntervalData(0, 1)
show(i)

## Show an interval-valued data list
list <- IntervalList(c(0, 3, 2, 5, 6), c(4, 5, 4, 8, 7))
show(list)

## Show an interval-valued data matrix
m <- IntervalMatrix(matrix(c(0, 1, 2, 3, 0, 3, 4, 9), 2, 4))
show(m)
```

`simulIVS` *Simulation of interval-valued responses to a questionnaire*

Description

This function allows to generate n interval-valued responses to each of k items of a questionnaire. These interval-valued responses are simulated mimicking the human behavior, considering three different procedures as it is suggested by De la Rosa de Sáa *et al*. (2015) and Lubiano *et al*. (2021).

Usage

```
simulIVS(
  n,
  k,
  w1 = 0.05,
  w2 = 0.35,
  w3 = 0.6,
  p = 2,
  q = 2,
  minimum = 1,
  maximum = 7
)
```

Arguments

n	A single positive integer indicating the number of different respondents that have answered to the questionnaire stored as a unique numeric object.
k	A single positive integer indicating the number of different items included in the questionnaire stored as a unique numeric object.
w1	A single real number in $[0, 1]$ indicating the proportion of observations that are generated in the first procedure stored as a unique numeric object. By default, w1 = 0.05.
w2	A single real number in $[0, 1]$ indicating the proportion of observations that are generated in the second procedure saved as a unique numeric object. By default, w2 = 0.35.
w3	A single real number in $[0, 1]$ indicating the proportion of observations that are generated in the third procedure stored as a unique numeric object. By default, w3 = 0.60.
p	A single non-negative real number which indicates the first parameter of a beta distribution. By default, p = 2.
q	A single non-negative real number which indicates the second parameter of a beta distribution. By default, q = 2.
minimum	A single real number indicating the lower bound of the interval-valued scale used saved as a unique numeric value. By default, minimum = 1.
maximum	A single real number indicating the upper bound of the interval-valued scale used saved as a unique numeric value. By default, maximum = 7.

Value

This function returns n interval-valued responses to each of k items in a questionnaire contained in a data.frame with n rows and $2k$ columns with values in the reference interval [minimum, maximum]. All interval-valued data's lower bounds appear in the first k columns of the data.frame and then all the corresponding upper bounds appear too.

Author(s)

José García-García <garciagarjose@uniovi.es>,
with contributions from María Asunción Lubiano <lubiano@uniovi.es>

References

- De la Rosa de Sáa, S.; Gil, M.Á.; González-Rodríguez, G.; López, M.T.; Lubiano M.A. (2015). Fuzzy rating scale-based questionnaires and their statistical analysis, *IEEE Transactions on Fuzzy Systems*, 23(1):111-126. doi:10.1109/TFUZZ.2014.2307895.
- Lubiano, M.A.; García-Izquierdo, A.L.; Gil, M.Á. (2021). Fuzzy rating scales: Does internal consistency of a measurement scale benefit from coping with imprecision and individual differences in psychological rating? *Information Sciences*, 550:91-108. doi:10.1016/j.ins.2020.10.042.

Examples

```
## Simulation some interval-valued responses to a questionnaire
## carried out under the following particular conditions
## Number of respondents: n = 100
## Number of items: k = 5
## Procedures proportions: (w1, w2, w3) = (0.10, 0.25, 0.65)
## Beta distribution parameters: (p, q) = (1, 7)
## Reference interval of interval-valued scales: [0, 10]
data <- simulIVS(100, 5, 0.10, 0.25, 0.65, 1, 7, 0, 10)
head(data)
```

---

spr *Extract and replace the spreads of interval-valued objects*

Description

This functions provides customized access to the `spr` slot of `IntervalData`, `IntervalList`, and `IntervalMatrix` objects, so their spreads can be extracted and replaced. This does not prevent to use the @ accessor, but does not force others to know the implementation details.

Usage

```
## S4 method for signature 'IntervalData'
spr(object)

## S4 method for signature 'IntervalList'
spr(object)

## S4 method for signature 'IntervalMatrix'
spr(object)

## S4 replacement method for signature 'IntervalData'
spr(object) <- value

## S4 replacement method for signature 'IntervalList'
spr(object) <- value

## S4 replacement method for signature 'IntervalMatrix'
spr(object) <- value
```

Arguments

`object`	A single nonempty compact interval or a list or matrix with some intervals of this family, that is, an `IntervalData`, `IntervalList`, or `IntervalMatrix` instance.
`value`	A `numeric` or `matrix` object with the new values of the intervals spreads.

Value

On the one hand, `spr()` function returns the spreads of the intervals contained in `IntervalData`, `IntervalList`, or `IntervalMatrix` instances stored as a single `numeric` value, a `numeric` vector, or a `matrix` object, respectively. On the other hand, `spr<-` command does not return any value since it only allows to replace the `spr` slot of the given interval-valued object.

Author(s)

José García-García <garciagarjose@uniovi.es>

See Also

For accessing `mid` slot of interval-valued objects, `mid()` function can be used.

Examples

```
## Some mid() function examples

## With IntervalData
interval <- IntervalData(0, 1)

spr(interval)
spr(interval) <- 0.75
spr(interval)

## With IntervalList
list <- IntervalList(c(1, 3, 5), c(2, 4, 6))

spr(list[c(1, 3)])
spr(list[c(1, 3)]) <- c(1, 3)
spr(list[c(1, 3)])

## With IntervalMatrix
m <- IntervalMatrix(matrix(c(1, 5, 2, 6, 6, 2,
                             7, 3, 3, 4, 4, 5), 2, 6))

spr(m[1, 1])
spr(m[1, 1]) <- 2
spr(m[1, 1])

spr(m[1, 1:2])
spr(m[1, 1:2]) <- c(2, 3)
spr(m[1, 1:2])

spr(m[, c(1, 3)])
spr(m[, c(1, 3)]) <- matrix(1:4, 2, 2)
spr(m[, c(1, 3)])
```

sum *Calculate the sum of n intervals*

Description

This function calculates the sum of n nonempty compact real intervals.

Usage

```
## S4 method for signature 'IntervalList'
sum(x)
```

Arguments

x	A list of intervals stored as an `IntervalList` object.

Details

This function generalizes the Minkowski's sum of two nonempty compact real intervals explained in arithmetic section and implemented through + operator's method for two `IntervalData` objects.

Value

This function returns an `IntervalData` object with the calculated sum of the given n intervals, which is defined as another nonempty compact real interval.

Author(s)

José García-García <garciagarjose@uniovi.es>

References

Hankin, R.K.S. (2010). A step-by-step guide to writing a simple package that uses S4 methods: a "hello world" example. Technical Report. Auckland University of Technology.

See Also

For further information of the interval arithmetic see arithmetic.

Examples

```
## The following code calculates the sum
## of a list with two different intervals
list <- IntervalList(c(1, 3), c(2, 5))
sum(list)
```

var *Calculate the sample Fréchet variance of a random interval*

Description

This function calculates the sample Fréchet variance of a single realization of n nonempty compact real intervals drawn from an interval-valued random set stored as an `IntervalList` object.

Usage

```
## S4 method for signature 'IntervalList'
var(x, theta = 1)
```

Arguments

`x`	A list of intervals, that is, an `IntervalList` object.
`theta`	A single positive real number saved as a `numeric` object. By default, `theta = 1`.

Details

Let $\mathcal{X}$ be an interval-valued random set and let $(x_1, x_2, \ldots, x_n)$ be a sample of n independent observations drawn from $\mathcal{X}$. Then, the sample Fréchet variance (see Fréchet, 1948) is defined as the following non-negative real number given by

$$s^2_{\mathcal{X}} = \frac{1}{n}\sum_{i=1}^{n} d^2_\theta\left(x_i, \overline{x}\right),$$

where $\theta > 0$ and $\overline{x}$ denotes the sample Aumann mean of $(x_1, x_2, \ldots, x_n)$. Due to θ-distance definition, this deviation measure can also be computed as follows,

$$s^2_{\mathcal{X}} = s^2_{\text{mid }\mathcal{X}} + \theta \cdot s^2_{\text{spr }\mathcal{X}},$$

where

$$s_{\text{mid }\mathcal{X}} = \frac{1}{n}\sum_{i=1}^{n}(\text{mid } x_i - \text{mid } \overline{x})^2,$$

$$s_{\text{spr }\mathcal{X}} = \frac{1}{n}\sum_{i=1}^{n}(\text{spr } x_i - \text{spr } \overline{x})^2.$$

Value

This function returns the calculated sample Fréchet variance of the given n interval, which is defined as a non-negative real number. Therefore, the output of this function is a single `numeric` object.

Author(s)

José García-García <garciagarjose@uniovi.es>

References

Fréchet, M. (1948). Les éléments aléatoires de nature quelconque dans un espace distancié. *Annales de l'institut Henri Poincaré*, 10(4):215-310.

See Also

Other sample central tendency and covariance measures such as sample Aumann mean and sample covariance can be calculated through `mean()` and `cov()` functions, respectively.

Examples

```
## Some var() examples
list <- IntervalList(c(1, 3), c(2, 5))
var(list)
var(list, theta = 1/3)
```

Index