

# Package ‘agrmt’

November 22, 2023

**Type** Package

**Title** Calculate Concentration and Dispersion in Ordered Rating Scales

**Version** 1.42.12

**Date** 2023-11-07

**Author** Didier Ruedin [aut, cre]  
Clem Aeppli [ctb]

**Maintainer** Didier Ruedin <didier.ruedin@unine.ch>

## Description

Calculates concentration and dispersion in ordered rating scales. It implements various measures of concentration and dispersion to describe what researchers variably call agreement, concentration, consensus, dispersion, or polarization among respondents in ordered data. It also implements other related measures to classify distributions. In addition to a generic city-block based concentration measure and a generic dispersion measure, the package implements various measures, including van der Eijk's (2001) <DOI:10.1023/A:1010374114305> measure of agreement A, measures of concentration by Leik, Tatsle and Wierman, Blair and Lacy, Kvalseth, Berry and Mielke, Reardon, and Garcia-Montalvo and Reynal-Querol. Furthermore, the package provides an implementation of Galtungs AJUS-system to classify distributions, as well as a function to identify the position of multiple modes.

**URL** <http://agrmt.r-forge.r-project.org>

**License** MIT + file LICENSE

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-11-22 17:30:02 UTC

## R topics documented:

agrmt-package . . . . .	2
agreement . . . . .	3
agreementError . . . . .	4
ajus . . . . .	6
ajusCheck . . . . .	7
ajusPlot . . . . .	8

BerryMielke . . . . .	9
BlairLacy . . . . .	10
cancel . . . . .	11
collapse . . . . .	11
compareAgreement . . . . .	13
compareValues . . . . .	14
concentration . . . . .	14
consensus . . . . .	15
consensus.variance . . . . .	16
D.variance . . . . .	17
disper . . . . .	18
dispersion . . . . .	18
dsquared . . . . .	19
entropy . . . . .	20
expand . . . . .	21
isd . . . . .	22
Kvalseth . . . . .	23
l.variance . . . . .	24
Leik . . . . .	25
lsquared . . . . .	26
lsquared.variance . . . . .	27
minnz . . . . .	28
modes . . . . .	28
MRQ . . . . .	29
patternAgreement . . . . .	30
patternVector . . . . .	31
polarization . . . . .	32
Reardon . . . . .	33
reduceVector . . . . .	34
sd.variance . . . . .	34
secondModes . . . . .	35
truncatevector . . . . .	36
var.variance . . . . .	37
<b>Index</b>	<b>38</b>

---

agrmt-package

*Calculate Concentration and Dispersion in Ordered Rating Scales*


---

## Description

This package calculates concentration and dispersion in ordered rating scales. It implements various measures of concentration and dispersion to describe what researchers variably call agreement, concentration, consensus, dispersion, or polarization among respondents in ordered data. It also implements other related measures to classify distributions.

## Details

In ordered rating scales, concentration occurs if many values on the scale cluster around one value. Dispersion describes the absence of concentration. Various measures exist to calculate concentration and dispersion.

The package provides a generic city-block based ([concentration](#)) measure, and a generic measure of dispersion ([disper](#)). To use Van der Eijk's (2001) algorithmic approach agreement "A", call [agreement](#). The derived [polarization](#) lets you calculate a polarization score based on agreement A. Values are inverted and standardized to [0, 1]. Other specific measures: Leik's measure of ordinal dispersion ([Leik](#)), Tatsle and Wierman's ([consensus](#)), Blair and Lacy's ([dsquared](#), [lsquared](#), and [BlairLacy](#)), the measure by Kvalseth ([Kvalseth](#)), Berry and Mielke's IOV ([BerryMielke](#)), Reardon ([Reardon](#)) or Garcia-Montalvo and Reynal-Querol's ([MRQ](#)).

The package also includes functions to classify distributions according to Galtung's (1969) AJUS-system ([ajus](#)), and changes over time according to Galtung's (1969) ISD-system ([isd](#)). Moreover, the function [modes](#) can identify the position of multiple modes.

## Author(s)

Didier Ruedin

Contributor: Clem Aepli

Maintainer: Didier Ruedin <didier.ruedin@unine.ch>

## References

van der Eijk, C. (2001) Measuring agreement in ordered rating scales, *Quality and Quantity* 35(3):325-341.

Galtung, J. (1969) *Theory and Methods of Social Research*. Oslo: Universitetsforlaget.

---

agreement

*Calculate van der Eijk's measure of agreement A*

---

## Description

Calculate agreement in ordered rating scales. This function implements van der Eijk's (2001) measure of agreement A, which can be used to describe agreement or consensus among respondents.

## Usage

```
agreement(V, old = FALSE)
```

## Arguments

V	A frequency vector
old	Optional argument if you wish to use the deprecated algorithm for agreement A, as outlined in van der Eijk's article. There is normally no reason to set the old argument.

## Details

This is the main function to calculate agreement. A frequency vector describes the number of observations in a given category. For example, the vector [10,20,30,15,4] describes 10 observations with position 1, 20 observations with position 2, 30 observations with position 3, 15 observations with position 4, and 4 observations with position 5. At least three categories are required to calculate agreement.

Polarization can be measured by extension. A convenience function [polarization](#) is provided.

## Value

The function returns the measure of agreement A. A is 1 if there is perfect unimodality (=agreement); A is 0 if there is perfect uniformity; A is -1 if there is perfect bimodality (=lack of agreement)

## Author(s)

Didier Ruedin

## References

van der Eijk, C. 2001. Measuring agreement in ordered rating scales. *Quality and Quantity* 35(3):325-341. <DOI: 10.1023/A:1010374114305>

## See Also

[polarization](#)

## Examples

```
# Sample data
V <- c(30,40,210,130,530,50,10)
# Calculate agreement A
agreement(V)
# The rate of agreement is given as 0.6113333
```

---

agreementError

*Simulated coding error for agreement A*

---

## Description

Calculate agreement in ordered rating scales, but simulates coding error.

## Usage

```
agreementError(V, n=500, e=0.01, pos=FALSE)
```

**Arguments**

V	A vector with an entry for each individual, not a frequency vector
n	Number of samples in the simulation
e	Proportion of samples for which errors are simulated
pos	Vector of possible positions. If FALSE (default), the values occurring in V are set as the possible values

**Details**

This function calculates agreement A, but simulates coding error. This can be useful to estimate standard errors and central tendency if certain positions are not observed. If all positions are observed in the vector V, bootstrapping can be used to estimate standard errors. If certain positions are not observed, bootstrapping is limited. Take an extreme example: [3 0 0 0 0]. Here we have three observations at the first position, but none at the others. Bootstrapping will always lead to the same agreement score. This can be misleading if coding error can be assumed. For example, if these three observations refer to a ‘strongly agree’ answer, it is usually conceivable that some or all of these answers could refer to ‘somewhat agree’. This function lets you specify how many of the observations should be assumed to be potentially mis-coded, and calculates agreement accordingly. If an observation is assumed to be potentially mis-coded, it is randomly set to the position to the left, the position to the right, or the position itself. If the first or last observation is chosen, the simulation takes care not to suggest values that could not occur.

You can run the function a few (hundred) times to get summary statistics of the result (mean, median, standard deviation, etc.). The function [compareAgreement](#) does just this, and compares the result with the agreement score if no coding error is assumed.

**Value**

The function returns the measure of agreement A.

**Author(s)**

Didier Ruedin

**See Also**

[agreement](#), [compareAgreement](#), [collapse](#)

**Examples**

```
# Sample data:
V <- c(1,1,1,1,2,3,3,3,3,4,4,4,4,4,4)
# calculate agreement; using collapse() to create the frequency vector
agreement(collapse(V))
# Calculate agreement A with coding error:
agreementError(V)
# Assume that all values could have coding error:
agreementError(V, e=1)
# Run the function a few times and show the mean:
z <- replicate(1000, agreementError(V))
```

```

mean(z)
hist(z) # etc.
# the example mentioned, population vector [3 0 0 0 0]:
V2 <- c(1,1,1)
agreementError(V2, pos=1:5)
# you can also use the compareAgreement function
compareAgreement(V2, pos=1:5)

```

---

ajus

*Classify distributions*


---

### Description

Classify distributions using the AJUS-system introduced by Galtung (1969).

### Usage

```
ajus(V, tolerance=0.1, variant="modified")
```

### Arguments

V	A frequency vector
tolerance	Specify how similar values have to be to be treated as different (optional). Differences smaller than or equal to the tolerance are ignored.
variant	Strict AJUS following Galtung, or modified to include F and L types (default)

### Details

This function implements the AJUS-system introduced by Galtung (1969). The input is a frequency vector; the output is a classification of the distribution.

Distributions are classified as A if they are unimodal with a peak in the centre, as J if they are unimodal with a peak at either end, as U if they are bimodal with a peak at both ends, and as S if they are multimodal. In addition to Galtung's classification, the function classifies distributions as F if there is no peak and all values are more or less the same (flat). Furthermore, a distinction is drawn between J and L distributions, depending on whether they increase or decrease: J types have a peak on the right, L types have the peak on the left. The skew is given as +1 for a positive skew, as 0 for no skew, and -1 for a negative skew.

The skew is identified by comparing the sum of values left and right of the midpoint respectively. For J-type of distributions, the skew is identified on the basis of the changes between values. This way, long tails cannot influence the skew, and a single peak at the left and right-hand end can be differentiated in all cases.

The aim of the AJUS system is to reduce complexity. Initially the intuition was to classify distributions on an ad-hoc basis (i.e. eye-balling). Using an algorithm is certainly more reliable, and useful if one is interested in classifying (and comparing) a large number of distributions. The argument `tolerance`, however is not a trivial choice and can affect results. Use the helper function `ajusCheck` to check sensitivity to different values of the tolerance parameter.

You can choose between a strict AJUS classification and a modified AJUSFL classification (default). The AJUS classification does not include a type for distributions without peaks (F type), and NA is returned instead. The AJUS classification does not draw a distinction between unimodal distributions with a peak at the end: the skew needs to be considered to distinguish between increasing and decreasing cases. The modified variant (default) includes the F type and the L type along with the original AJUS types.

### Value

The function returns a list. The type returns a string corresponding to the pattern described by Galtung (A,J,U,S) or (F,L). The skew returns a number to describe the direction of the skew. The pattern returns the simplified pattern of the distribution. It indicates whether two values were considered the same (0), or if there was an increase (1) or decrease (-1) between two consecutive values. The length of the pattern is equal to the length of the frequency vector minus one.

### Author(s)

Didier Ruedin

### References

Galtung, J. (1969) Theory and Methods of Social Research. Oslo: Universitetsforlaget.

### See Also

[isd](#), [ajusCheck](#), [ajusPlot](#)

---

ajusCheck

*Sensitivity test for AJUS*

---

### Description

Check sensitivity of AJUS to different tolerance parameters.

### Usage

```
ajusCheck(V, t=seq(from=0.05, to=0.2, by=0.05), variant="modified")
```

### Arguments

V	A frequency vector
t	A vector of tolerance parameters to check. Differences smaller than or equal to the tolerance are ignored.
variant	Strict AJUS following Galtung, or modified to include F and L types (default)

### Details

This function runs the AJUS system with a range of tolerance parameters. This way, you can easily check how sensitive the classification of the distribution is to the tolerance parameter.

**Value**

The function returns a list. The tolerance returns the tolerance parameters tested. The type returns a series of strings corresponding to the pattern described by Galtung (A,J,U,S) or (F, L) for each tolerance parameter. The skew returns a number to describe the direction of the skew. See [ajus](#) for a description of the different arguments and the AJUS types.

**Author(s)**

Didier Ruedin

**See Also**

[ajus](#)

---

ajusPlot

*Plot vector with AJUS type*

---

**Description**

Plot a frequency vector among with its AJUS type.

**Usage**

```
ajusPlot(V, tolerance=0.1, variant="modified", ...)
```

**Arguments**

V	A frequency vector
tolerance	Specify how similar values have to be to be treated as different (optional). Differences smaller than or equal to the tolerance are ignored.
...	Arguments to pass to the plotting function
variant	Strict AJUS following Galtung, or modified to include F and L types (default)

**Details**

This function plots the frequency vector along with its AJUS classification and skew. See [ajus](#) for a description of the AJUS system and the different parameters. In contrast to the `ajus` function, `ajusPlot` can deal with missing values. Missing values are removed when calculating the AJUS type (because AJUS does not handle missing values), but they are considered in the plot. This makes `ajusPlot` useful for classifying time series where missing values may occur. Additional arguments can be passed to the underlying plot function.

**Author(s)**

Didier Ruedin



**See Also**[ajus](#)

---

**BerryMielke***Calculate IOV*

---

**Description**

Calculate Berry and Mielke's IOV.

**Usage**

```
BerryMielke(V)
```

**Arguments**

V                    A frequency vector

**Details**

This function calculates Berry and Mielke's IOV, a measure of dispersion based on squared Euclidean distances. This function follows the presentation by Blair and Lacy 2000, but includes the adjustment for Tmax omitted by Blair and Lacy as there is no reason to leave it out. The derived measure COV by Kvalseth is implemented as [Kvalseth](#). Usually, the IOV is equivalent to [1-lsquared](#).

**Value**

The function returns the IOV.

**Author(s)**

Didier Ruedin

**References**

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. *Sociological Methods & Research* 28 (3): 251-280.

Berry, K., and P. Mielke. 1992. Assessment of Variation in Ordinal Data. *Perceptual and Motor Skills* 74 (1): 63-66.

**See Also**

[1squared](#), [Kvalseth](#)

**Examples**

```
# Sample data
V <- c(30,40,210,130,530,50,10)
BerryMielke(V)
```

---

BlairLacy

*Calculate l*

---

**Description**

Calculate Blair and Lacy's  $l$ .

**Usage**

```
BlairLacy(V)
```

**Arguments**

$V$                     A frequency vector

**Details**

This function calculates Blair and Lacy's  $l$ , a measure of concentration based on linear Euclidean distances. This function follows the presentation by Blair and Lacy 2000. The measure  $l$ -squared by Blair and Lacy is implemented as [lsquared](#).

**Value**

The function returns the  $l$ .

**Author(s)**

Didier Ruedin

**References**

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. *Sociological Methods & Research* 28 (3): 251-280.

**See Also**

[lsquared](#)

**Examples**

```
# Sample data
V <- c(30,40,210,130,530,50,10)
BlairLacy(V)
```

---

censor	<i>Censor helper function</i>
--------	-------------------------------

---

**Description**

Helper function to censor frequency vectors

**Usage**

```
censor(V, left=0, right=100)
```

**Arguments**

V	Vector
left	left to censor
right	right to censor

**Details**

Helper function to censor frequency vectors

**Value**

censored frequency vector

**Author(s)**

Clem Aepli

**See Also**

[truncate](#)

---

collapse	<i>Reduces a vector to a frequency vector</i>
----------	---

---

**Description**

Takes a vector and reduces it to a frequency vector.

**Usage**

```
collapse(D, pos=FALSE, na.rm=TRUE)
```

## Arguments

D	Vector
pos	Optional: position of categories
na.rm	Optional: should NA be removed (TRUE by default)

## Details

This function reduces a vector to a frequency vector. This function is very similar to the way `table` summarizes vectors, but this function can deal with categories of frequency 0 if the argument `pos` is specified. Here we assume a vector with an entry for each individual (the order of the values is ignored). Each entry states the position of an individual. When the number of positions is naturally limited, such as when categorical positions are used, frequency vectors can summarize this information: how many individuals have position 1, how many individuals have position 2, etc. A frequency vector has an entry for each position in the population (sorted in ascending order). Each entry states the number of individuals in the population with this position.

The argument `pos` is required if certain positions do not occur in the population (or if there is a chance that they do not occur in a specific sub-population). For example, if we have positions on a 7-point scale, and position 3 never occurs in the population, the argument `pos` must be specified. In this case, the argument may be `pos=1:7`. We can also use categories more generally, as in `c(-3, -1, 0, 0.5, 1, 2, 5)`. Specifying the positions of categories when all positions occur in the population has no side-effects. See the example for an illustration.

By default, missing values (NA) are removed with `as.numeric(na.omit())`. This helps with some vectors that include NA that fail otherwise. If NA are maintained with `na.rm=FALSE`, they are included as the last category. The argument `pos` cannot include NA as a position; NA are removed if the argument `pos` is used.

## Value

A frequency vector

## Author(s)

Didier Ruedin

## See Also

[expand](#)

## Examples

```
V = c(1,1,1,1,1,1,3,3,3,3,4,5,5,5,5)
# summarize using table()
table(V)
# summarize using collapse()
collapse(V)
# assuming possible values (1,2,3,4,5), we get that zero times 2 included:
collapse(V, pos=c(1,2,3,4,5))
```

---

compareAgreement	<i>Compare agreement A with and without simulated coding error</i>
------------------	--

---

### Description

Calculate agreement in ordered rating scales, and compares this to agreement with simulated coding error.

### Usage

```
compareAgreement(V, n=500, e=0.01, N=500, pos=FALSE)
```

### Arguments

V	A vector with an entry for each individual
n	Number of samples in the simulation of coding errors
e	Proportion of samples for which errors are simulated
N	Number of replications for calculating mean and standard deviation
pos	Vector of possible positions. If FALSE, the values occurring in V are set as the possible values

### Details

This function calculates agreement on a vector, and compares the value with agreement with simulated coding error. It runs the function [agreementError](#) N times. The other arguments (n, e, pos) are passed down to the [agreementError](#) function.

### Value

The function returns a list with agreement A without simulated coding errors, the mean of agreement with simulated coding error, and the standard deviation of agreement with simulated coding error.

### Author(s)

Didier Ruedin

### See Also

[agreement](#), [agreementError](#)

### Examples

```
# Sample data:  
V <- c(1,1,1,1,2,3,3,3,3,4,4,4,4,4,4)  
compareAgreement(V)
```

---

compareValues	<i>Compares two values</i>
---------------	----------------------------

---

**Description**

This is a helper function to compare two values.

**Usage**

```
compareValues(A,B,tolerance=0.1)
```

**Arguments**

A	A number
B	A number
tolerance	Specify how similar values have to be to be treated as different. Differences smaller than or equal to the tolerance are ignored.

**Details**

This is a helper function compare two values. Two values are more or less the same, or one of the two is bigger.

**Value**

The function returns number to describe the relationship: -1 if A is bigger, 1 if B is bigger, and 0 if the two are more or less the same.

**Author(s)**

Didier Ruedin

---

concentration	<i>Measures concentration</i>
---------------	-------------------------------

---

**Description**

This function measures concentration in a frequency vector

**Usage**

```
concentration(V, metric=2)
```

**Arguments**

V	Vector
metric	city block metric

**Details**

This function measures concentration in a frequency vector in city blocks. The default metric is 2.

**Value**

measure of concentration

**Author(s)**

Clem Aeppli

**Examples**

```
# Sample data
V <- c(30,40,210,130,530,50,10)
concentration(V)
```

---

consensus

*Calculate Tastle and Wierman's measure of consensus*

---

**Description**

Calculate consensus in ordered rating scales. This function implements Tastle and Wierman's (2007) measure of consensus (ordinal dispersion), which can be used to describe agreement, consensus, dispersion, or polarization among respondents.

**Usage**

```
consensus(V)
```

**Arguments**

V	A frequency vector
---	--------------------

**Details**

This function calculates consensus following Tastle and Wierman (2007). The measure of consensus is based on the Shannon entropy. A frequency vector describes the number of observations in a given category. For example, the vector [10,20,30,15,4] describes 10 observations with position 1, 20 observations with position 2, 30 observations with position 3, 15 observations with position 4, and 4 observations with position 5.

If you come across an error that the vector supplied does not contain whole numbers, try `round(V, 0)` to remove any detritus from calculating the frequency vector.

**Value**

The function returns the measure of consensus. It is 1 if there is perfect uniformity; it is 0 if there is perfect bimodality (=lack of agreement)

**Author(s)**

Didier Ruedin

**References**

Tastle, W., and M. Wierman. 2007. Consensus and dissention: A measure of ordinal dispersion. *International Journal of Approximate Reasoning* 45(3): 531-545.

**See Also**

[agreement](#)

**Examples**

```
# Sample data
V <- c(30,40,210,130,530,50,10)
# Calculate consensus
consensus(V)
# The degree of consensus is given as 0.7256876
```

---

consensus.variance      *Approximate variance of consensus*

---

**Description**

Calculate approximate variance of Blair and Lacy's consensus Cns

**Usage**

```
consensus.variance(V)
```

**Arguments**

V                      Frequency vector

**Details**

Helper function to calculate approximate variance of Blair and Lacy's (2000) consensus Cns.

**Value**

Approximate variance of Blair and Lacy's consensus Cns



**Author(s)**

Clem Aeppli

**References**

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. Sociological Methods & Research 28 (3): 251-280.

**See Also**

[consensus](#)

---

D.variance

*Approximate variance of Leik's D*

---

**Description**

Calculate approximate variance of Leik's D

**Usage**

D.variance(V)

**Arguments**

V                      Frequency vector

**Details**

Helper function to calculate approximate variance of Leik's (1966) D.

**Value**

Approximate variance of D

**Author(s)**

Clem Aeppli

**References**

Leik, R. (1966) A measure of ordinal consensus, Pacific Sociological Review 9(2):85-90.

**See Also**

[Leik](#)

---

disper	<i>Measures distance</i>
--------	--------------------------

---

**Description**

This function measures distance between two frequency vectors

**Usage**

```
disper(A, B, metric=2)
```

**Arguments**

A	Vector
B	Vector
metric	city block metric

**Details**

Calculates the distance between two frequency vectors.

**Value**

measure of distance

**Author(s)**

Clem Aepli

**See Also**

[concentration](#)

---

dispersion	<i>Measures dispersion</i>
------------	----------------------------

---

**Description**

This function measures dispersion in a frequency vector

**Usage**

```
dispersion(V, metric=2)
```

**Arguments**

V	Vector
metric	city block metric

**Details**

dispersion is 1-concentration

**Value**

measure of dispersion

**Author(s)**

Clem Aeppli

**See Also**

[concentration](#)

---

dsquared

*Calculate d-squared*

---

**Description**

Calculate Blair and Lacy's d-squared.

**Usage**

dsquared(V)

**Arguments**

V	A frequency vector
---	--------------------

**Details**

This function calculates Blair and Lacy's d-squared, a measure of concentration based on squared Euclidean distances. This function follows the presentation by Blair and Lacy 2000. The measure 1-squared normalizes the values and is implemented as [lsquared](#).

**Value**

The function returns the d-squared.

**Author(s)**

Didier Ruedin

## References

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. *Sociological Methods & Research* 28 (3): 251-280.

## See Also

[lsquared](#), [BlairLacy](#)

## Examples

```
# Sample data
V <- c(30,40,210,130,530,50,10)
dsquared(V)
```

---

entropy

*Calculate Shannon entropy*

---

## Description

Calculate Shannon entropy, following Tastle and Wierman.

## Usage

```
entropy(V)
```

## Arguments

V                    A frequency vector

## Details

This function calculates the Shannon entropy following Tastle and Wierman (2007). A frequency vector describes the number of observations in a given category. For example, the vector [10,20,30,15,4] describes 10 observations with position 1, 20 observations with position 2, 30 observations with position 3, 15 observations with position 4, and 4 observations with position 5.

This function follows Tastle and Wierman and ignores categories with zero observations. This does not follow the formula indicated.

See [consensus](#) for a function that considers the order of categories.

## Value

The function returns the Shannon entropy.

## Author(s)

Didier Ruedin

**References**

Tastle, W., and M. Wierman. 2007. Consensus and dissent: A measure of ordinal dispersion. *International Journal of Approximate Reasoning* 45 (3): 531-545.

**See Also**

[consensus](#)

**Examples**

```
# Sample data
V <- c(30,40,210,130,530,50,10)
# Calculate entropy
entropy(V)
```

---

expand

*Expands a frequency vector to a vector*

---

**Description**

This function expands a frequency vector to a vector.

**Usage**

```
expand(F)
```

**Arguments**

F                      Frequency vector

**Details**

This function takes a frequency vector and expands it to a longer vector with one entry for each observation. It is reverses the [collapse](#) function. A frequency vector has an entry for each position in the population. Each entry states the number of individuals in the population with this position. Here we create a vector with an entry for each individual.

**Value**

A vector

**Author(s)**

Didier Ruedin

**See Also**

[collapse](#)

---

`isd`*Classify changes over time*

---

**Description**

Classify changes over time using the ISD-system introduced by Galtung (1969).

**Usage**

```
isd(V, tolerance=0.1)
```

**Arguments**

<code>V</code>	A vector with length 3
<code>tolerance</code>	Specify how similar values have to be to be treated as different (optional). Differences smaller than or equal to the tolerance are ignored.

**Details**

This function implements the ISD-system introduced by Galtung (1969). The input is a vector of length 3. Each value stands for a different point in time. The ISD-system examines the two transition points, and classifies the changes over time.

**Value**

The function returns a list. The `type` returns a number corresponding to the pattern described by Galtung. The `description` returns a string where the two transitions are spelled out (increase, flat, decrease).

**Author(s)**

Didier Ruedin

**References**

Galtung, J. (1969) Theory and Methods of Social Research. Oslo: Universitetsforlaget.

**See Also**

[ajus](#)

---

`Kvalseth`*Calculate Kvalseth's COV*

---

**Description**

Calculate Kvalseth's COV.

**Usage**

```
Kvalseth(V)
```

**Arguments**

`V`                    A frequency vector

**Details**

This function calculates Kvalseth's COV, a measure of dispersion based on linear Euclidean distances. It is based on the IOV measure, implemented as [BerryMielke](#). This function follows the presentation by Blair and Lacy 2000.

**Value**

The function returns the COV.

**Author(s)**

Didier Ruedin

**References**

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. *Sociological Methods & Research* 28 (3): 251-280.

**See Also**

[BerryMielke](#), [lsquared](#)

**Examples**

```
# Sample data
V <- c(30,40,210,130,530,50,10)
Kvalseth(V)
```

---

l.variance	<i>Approximate variance of Blair and Lacy's l</i>
------------	---

---

**Description**

Calculate approximate variance of Blair and Lacy's (2000) l

**Usage**

```
l.variance(V)
```

**Arguments**

V                      Frequency vector

**Details**

Helper function to calculate approximate variance of Blair and Lacy's (2000) l.

**Value**

Approximate variance of Blair and Lacy's l

**Author(s)**

Clem Aeppli

**References**

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. *Sociological Methods & Research* 28 (3): 251-280.

**See Also**

[BlairLacy](#)



---

Leik

*Calculate ordinal dispersion*

---

### **Description**

Calculates ordinal dispersion as introduced by Leik (1966)

### **Usage**

Leik(V)

### **Arguments**

V                    A frequency vector

### **Details**

This function calculates ordinal dispersion as introduced by Robert K. Leik (1966). It uses the cumulative frequency distribution to determine ordinal dispersion. The extremes (agreement, polarization) largely correspond to the types used by Cees van der Eijk. By contrast, the mid-point depends on the number of categories: it tends toward 0.5 as the number of categories increases. Leik defends this difference by highlighting the increased probability of falling into polarized patterns when there are fewer categories. If all observations are in the same category, ordinal dispersion is 0. With half the observations in one extreme category, and half the observations in the other extreme, Leik's measure gives a value of 1.

The dispersion measure is a percentage, and can be interpreted accordingly. Ordinal dispersion can be used to express consensus or agreement, simply by taking: 1 - ordinal dispersion.

### **Value**

The function returns the ordinal dispersion

### **Author(s)**

Didier Ruedin

### **References**

Leik, R. (1966) A measure of ordinal consensus, *Pacific Sociological Review* 9(2):85-90.

### **See Also**

[polarization, agreement](#)

**Examples**

```
# Example 1:
V <- c(30,40,210,130,530,50,10)
# Calculate polarization
Leik(V)
# The ordinal dispersion is given as 0.287
polarization(V)
# Polarization is given as 0.194 (as contrast)
```

---

Isquared

*Calculate l-squared*

---

**Description**

Calculate Blair and Lacy's l-squared.

**Usage**

```
Isquared(V)
```

**Arguments**

V                    A frequency vector

**Details**

This function calculates Blair and Lacy's l-squared, a measure of concentration based on squared Euclidean distances. This function follows the presentation by Blair and Lacy 2000. The measure 'l' by Blair and Lacy is implemented as [BlairLacy](#).

**Value**

The function returns the l-squared.

**Author(s)**

Didier Ruedin

**References**

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. *Sociological Methods & Research* 28 (3): 251-280.

**See Also**

[BerryMielke](#), [BlairLacy](#)

**Examples**

```
# Sample data
V <- c(30,40,210,130,530,50,10)
lsquared(V)
```

---

lsquared.variance      *Approximate variance of Blair and Lacy's Isquared*

---

**Description**

Calculate approximate variance of Blair and Lacy's (2000) Isquared

**Usage**

```
lsquared.variance(V)
```

**Arguments**

V                      Frequency vector

**Details**

Helper function to calculate approximate variance of Blair and Lacy's (2000) Isquared.

**Value**

Approximate variance of Blair and Lacy's Isquared

**Author(s)**

Clem Aepli

**References**

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. *Sociological Methods & Research* 28 (3): 251-280.

**See Also**

[BlairLacy](#)

---

minnz	<i>Non-zero minimum</i>
-------	-------------------------

---

**Description**

Helper function to calculate the smallest value of the vector except for 0 (non-zero minimum).

**Usage**

```
minnz(V)
```

**Arguments**

V	A vector
---	----------

**Details**

This is a helper function to calculate the non-zero minimum of a vector. The result is the smallest value of the vector, but cannot be zero.

**Value**

The function returns the non-zero minimum

**Author(s)**

Didier Ruedin

---

modes	<i>Identify multiple modes</i>
-------	--------------------------------

---

**Description**

Identifies (multiple) modes in a frequency vector.

**Usage**

```
modes(V, pos=FALSE, tolerance=0.1)
```

**Arguments**

V	A frequency vector
pos	Categories of frequency vector (optional)
tolerance	Specify how similar values have to be to be treated as different (optional). Differences smaller than or equal to the tolerance are ignored.

## Details

This function identifies which positions of a frequency vector correspond to the mode. If there are multiple modes of the same value, all matching positions will be reported. Use the function [collapse](#) to create frequency vectors if necessary.

## Value

The function returns a list. The `at` returns the categories of the frequency vector. Either these categories were specified using the argument `pos`, or we assume it to be `1:k` (with `k` the number of categories in the frequency vector). If the length of the `pos` argument does not match the length of the frequency vector, a warning is shown, and the `pos` argument is ignored. The `frequencies` returns the frequency vector. The `mode` returns the value of the mode(s). If there are multiple modes, they are listed. Similar frequencies are counted as equal, using the `tolerance` argument. To prevent similar frequencies to be considered the same, set `tolerance` to 0. The `positions` returns the positions of the vector that correspond to the mode. This will differ from the `mode` if `pos` is provided. The `contiguous` returns `TRUE` if all modes are contiguous, and `FALSE` if there are different values in between. If there is only one mode, it is defined as contiguous (i.e. `TRUE`).

## Author(s)

Didier Ruedin

## See Also

[secondModes](#)

## Examples

```
# Example 1: finding the mode
V1 <- c(30,40,210,130,530,50,10)
modes(V1) # will find position 5
# Example 2:
V2 <- c(3,0,4,1)
modes(V2) # will find position 3
# Example 3: providing categories
modes(V2,pos=-1:2) # will still find position 3, but give the value of 1 as mode
# Example 4: similar values
V3 <- c(30,40,500,130,530,50,10)
modes(V3, tolerance=30) # will find positions 3 and 5 (500 and 530 are nearly the same)
```

---

MRQ

*Calculates MRQ polarization index*

---

## Description

This function calculates the MRQ polarization index from a population vector.

**Usage**

```
MRQ(Z)
```

**Arguments**

Z (Standardized) frequency vector

**Details**

This function implements the polarization index introduced by Garcia-Montalvo and Reynal-Querol (2005), also known as the Reynal-Querol index of polarization (RQ). It is a measure of dispersion based on squared Euclidean distances. The frequency vector needs to be standardized for the Reynal-Querol index to work; if the sum of the frequency vector is not 1 (i.e. it is not standardized), the function automatically standardizes the frequency vector by dividing each element of the vector by the sum of the vector. The assumption is that the frequencies are complete.

**Value**

Index of polarization (RQ).

**Author(s)**

Didier Ruedin

**References**

Garcia-Montalvo, Jose, and Marta Reynal-Querol. 2005. Ethnic Polarization, Potential Conflict, and Civil Wars. *American Economic Review* 95(3): 796-816.

Reynal-Querol, Marta. 2002. Ethnicity, Political Systems, and Civil Wars. *Journal of Conflict Resolution* 46(1): 29-54.

**Examples**

```
# Sample data
V <- c(30,40,210,130,530,50,10)
MRQ(V)
```

---

patternAgreement	<i>Calculates patterns agreement</i>
------------------	--------------------------------------

---

**Description**

Helper function to calculate agreement A from a pattern vector.

**Usage**

```
patternAgreement(P, old=FALSE)
```

**Arguments**

P	A pattern vector
old	Optional argument if the old algorithm for agreement A is to be used. There is normally no reason to set the old argument.

**Details**

This is a helper function to calculate agreement A from a pattern vector.

**Value**

The function returns the measure of agreement A

**Author(s)**

Didier Ruedin

**See Also**

[agreement](#)

---

patternVector	<i>Creates pattern vector</i>
---------------	-------------------------------

---

**Description**

Helper function to create a pattern vector from a frequency vector.

**Usage**

```
patternVector(V)
```

**Arguments**

V	A frequency vector
---	--------------------

**Details**

This is a helper function to create a pattern vector from a frequency vector. A pattern vector reduced all values greater or equal to 1 to 1, and values of 0 remain 0. A frequency vector (0,0,18,59,0,34,2) is turned into a pattern vector (0,0,1,1,0,1,1).

**Value**

The function returns a pattern vector.

**Author(s)**

Didier Ruedin

**See Also**[agreement](#)

---

polarization

*Calculate polarization*

---

**Description**

Calculates polarization, based on measure of agreement A

**Usage**`polarization(V, old = FALSE)`**Arguments**

V	A frequency vector
old	Specify old=TRUE to use the deprecated algorithm for agreement A

**Details**

This function calculates polarization by re-scaling agreement A introduced by Cees van der Eijk. Whereas agreement A ranges from -1 to 1, polarization ranges from 0 to 1. If all observations are in the same category, polarization is 0. With half the observations in one category, and half the observations in a different (non-neighbouring) category, polarization is 1. Polarization is 0.5 for a uniform distribution over all categories.

**Value**

The function returns a polarization score

**Author(s)**

Didier Ruedin

**See Also**[agreement](#)**Examples**

```
V <- c(30,40,210,130,530,50,10)
# Calculate polarization
polarization(V)
# The rate of polarization is given as 0.1943333
```



---

Reardon	<i>Reardon's entropy</i>
---------	--------------------------

---

**Description**

This function calculates Reardon's (2009) entropy from a frequency vector

**Usage**

```
Reardon(V)
```

**Arguments**

V                      Frequency vector

**Details**

Calculate Reardon's (2009) entropy.

**Value**

measure of entropy

**Author(s)**

Clem Aeppli

**References**

Reardon, S. 2009. Measures of Ordinal Segregation. *Research on Economic Inequality* 17:129-55.  
<DOI: 10.1108/S1049-2585(2009)0000017011>

**See Also**

[disper](#)

---

reduceVector	<i>Remove zeros and repeated values</i>
--------------	---

---

**Description**

This is a helper function to remove all zeros and repeated values from a vector.

**Usage**

```
reduceVector(X)
```

**Arguments**

X                    A (frequency) vector

**Details**

This is a helper function to strip all zeros and repeated values from a vector.

**Value**

The function returns vector

**Author(s)**

Didier Ruedin

**See Also**

[agreement](#)

---

sd.variance	<i>Approximate variance of the categorical standard deviation</i>
-------------	---

---

**Description**

Calculate approximate variance of the categorical standard deviation

**Usage**

```
sd.variance(V)
```

**Arguments**

V                    Frequency vector

**Details**

Helper function to calculate approximate variance of the categorical standard deviation.

**Value**

Approximate variance of the categorical standard deviation

**Author(s)**

Clem Aepli

**See Also**

[truncate](#)

---

secondModes

*Most common and second most common values*

---

**Description**

Identifies the most common (multiple) modes for frequency vectors as well as the second most common values.

**Usage**

```
secondModes(V, pos=FALSE, tolerance=0.1)
```

**Arguments**

V	A frequency vector
pos	Categories of frequency vector (optional)
tolerance	Specify how similar values have to be to be treated as different (optional). Differences smaller than or equal to the tolerance are ignored.

**Details**

This function identifies which positions of a frequency vector correspond to the mode(s) as implemented in the [modes](#) function. It also reports the second most common position in the same manner.

**Value**

The function returns a list for the most common and the second most common value(s). The output corresponds to that of the [modes](#) function.

**Author(s)**

Didier Ruedin

**See Also**[modes](#)

---

truncatevector	<i>Truncate helper function</i>
----------------	---------------------------------

---

**Description**

Helper function to truncate frequency vectors

**Usage**

```
truncatevector(V, left=0, right=100)
```

**Arguments**

V	Vector
left	left to truncate
right	right to truncate

**Details**

Helper function to truncate frequency vectors

**Value**

truncated frequency vector

**Author(s)**

Clem Aepli

**See Also**[censor](#)

---

var.variance	<i>Approximate variance of the consensus (Cns) estimator</i>
--------------	--

---

**Description**

Calculate approximate variance of the consensus (Cns) estimator

**Usage**

```
var.variance(V)
```

**Arguments**

V                      Frequency vector

**Details**

Helper function to calculate approximate variance of the Consensus (Cns) estimator.

**Value**

Approximate variance of Blair and Lacy's I

**Author(s)**

Clem Aepli

**References**

Blair, J., and M. Lacy. 2000. Statistics of Ordinal Variation. *Sociological Methods & Research* 28 (3): 251-280.

**See Also**

[consensus](#)

# Index

## \* package

agrmt-package, 2

agreement, 3, 3, 5, 13, 16, 25, 31, 32, 34  
agreementError, 4, 13  
agrmt (agrmt-package), 2  
agrmt-package, 2  
ajus, 3, 6, 8, 9, 22  
ajusCheck, 7, 7  
ajusPlot, 7, 8

BerryMielke, 3, 9, 23, 26  
BlairLacy, 3, 10, 20, 24, 26, 27

sensor, 11, 36  
collapse, 5, 11, 21, 29  
compareAgreement, 5, 13  
compareValues, 14  
concentration, 3, 14, 18, 19  
consensus, 3, 15, 17, 20, 21, 37  
consensus.variance, 16

D.variance, 17  
disper, 3, 18, 33  
dispersion, 18  
dsquared, 3, 19

entropy, 20  
expand, 12, 21

isd, 3, 7, 22

Kvalseth, 3, 9, 23

l.variance, 24  
Leik, 3, 17, 25  
lsquared, 3, 9, 10, 19, 20, 23, 26  
lsquared.variance, 27

minnz, 28  
modes, 3, 28, 35, 36

MRQ, 3, 29

patternAgreement, 30  
patternVector, 31  
polarization, 3, 4, 25, 32

Reardon, 3, 33  
reduceVector, 34

sd.variance, 34  
secondModes, 29, 35

truncate, 11, 35  
truncatevector, 36

var.variance, 37