An Introduction to the R Package Agrmt

Didier Ruedin
University of Neuchâtel and University of the Witwatersrand
didier.ruedin@unine.ch

October 10, 2026

1 Overview

This package provides functions to calculate concentration and dispersion in ordered rating scales. Concentration may also be described as agreement, consensus; dispersion is also known as polarization. It also implements other related measures to classify distributions. At its core, 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 implemented: 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`) and Garcia-Montalvo and Reynal-Querol's (`MRQ`).

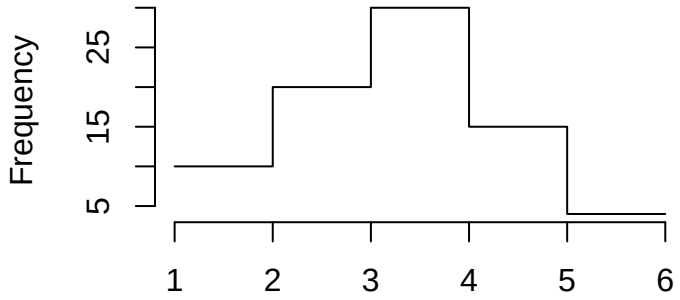
For more details on the different measures of agreement and how they compare, see: Aeppli, Clem, and Didier Ruedin. 2026. Let's Measure Concentration and Dispersion in Ordinal Data. Methods, Data, Analyses. Online First. `https://doi.org/10.25521/mda.578`.

The package 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.

2 Getting Data In

The functions in this package use what I refer to as frequency vectors. 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 (see Figure for a graphical

distribution of this frequency vector). At least three categories are required to calculate agreement.



There are many ways to create frequency vectors, including the `table` function in R. Your data may already come in the form of a frequency vector. The package provides a helper function `collapse` to cater with empty cells. Consider a simple case to illustrate the difference:

```
> library(agrmt)
> x = c(1,1,3) # these are our data
> # 2 observations with position 1,
> # 1 observation with position 3
> table(x)

x
1 3
2 1

> collapse(x)

[1] 2 1

> collapse(x, pos=1:3) # now we specify which categories exist

[1] 2 0 1
```

Using `table` we simply get a summary of the positions and number of observations at these positions. The helper function `collapse` lets us specify which categories actually exist. In this example, we specify that these are observations on a variable with 3 categories: 1, 2, and 3. The fact that there are *no* observations at position 2 is important information when calculating agreement.

Some measures require standardized frequency vectors, that is the frequencies expressed as a proportion. In this example, the proportion of responses at position 1 is 0.67, the proportion at position 2 is 0, and the proportion at position 3 is 0.33. Where standardized frequency vectors are required, frequency vectors that do not sum to 1 (i.e. that are not standardized) are automatically standardized by dividing each element of the frequency vector by the sum of the vector. This assumes complete data; a general assumption of the measures in this package.

3 Agreement

3.1 Concept: Agreement in Ordered Rating Scales

Ordered rating scales are a common format in surveys, and often we are interested in the extent to which responses are in agreement (whether there is consensus, whether there is concentration among the responses). Although frequently used, several researchers questioned whether standard deviations are appropriate in this case. Van der Eijk's measure of agreement 'A' is an algorithm that disaggregates frequency distributions into component parts called layers. At the level of these layers, agreement can easily be determined, and the measure of agreement provides the weighted average.

We can use the example provided by van der Eijk (p.331) to illustrate the use of layers.

Position on rating scale	1	2	3	4	5	6	7
Observed frequencies	30	40	210	130	530	50	10

The observed frequencies (30, 40, 210, 130, 530, 50, 10) constitute a frequency vector. The measure of agreement divides this into layers, starting with the lowest observed frequencies. The level of agreement for this layer is calculated, and weighted by the number of observations in this layer. This is repeated for all layers, using any additional observations, until we reach the highest observed frequencies. We begin with the cell with the lowest frequencies, in this case position 7 with 10 observations. This means that for the first layer, there are 10 observations for each position. This gives a level of agreement of 0 for this level. The weight of the level is the number of observations in this layer (70; that is, 7 [positions] times 10 [observations]) divided by the total number of observations (1000). In this example, the weight of the first layer is $\frac{70}{1000} = 0.07$. For the second layer, we have already used all 10 observations for position 7, so a 0 will be added. The second-lowest frequency is the 20 left for position 1 (30 original observations minus 10 in level 1). The level of agreement for the second layer is 0.17, with a weight of 0.12.

We continue this way until all observations are used:

The frequency vector is visible in the bottom row ('total'). To calculate the level of agreement of each layer, the pattern of this layer is considered. A pattern

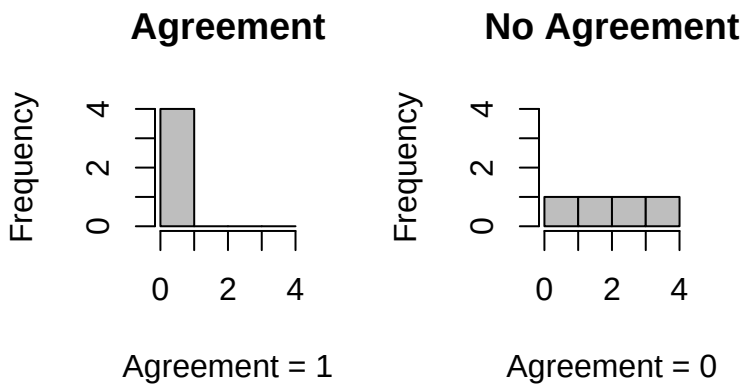
	1	2	3	4	5	6	7	Layer	Pattern	Weight
Layer 1	10	10	10	10	10	10	10	0	1111111	0.07
Layer 2	20	20	20	20	20	20	0	0.167	1111110	0.12
Layer 3	0	10	10	10	10	10	0	0.333	0111110	0.05
Layer 4	0	0	10	10	80	10	0	0.5	0011110	0.04
Layer 5	0	0	80	80	80	0	0	0.667	0011100	0.24
Layer 6	0	0	80	0	80	0	0	0.467	0010100	0.16
Layer 7	0	0	0	0	320	0	0	1	0000100	0.32
Total	30	40	210	130	530	50	10			

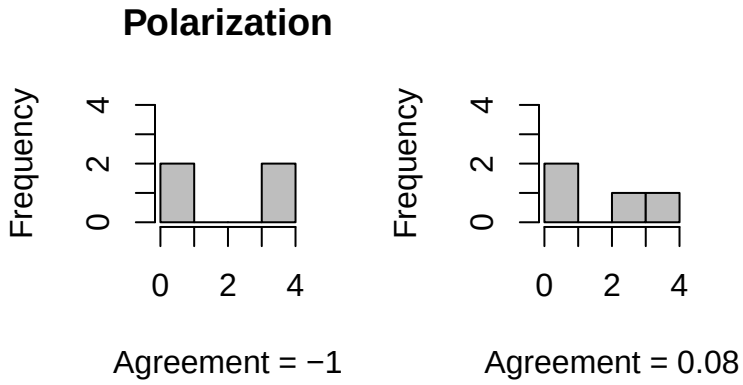
consists of 0 and 1. A 1 denotes any value other than 0 in the layer. For instance, in layer 2, all 20 are replaced with 1 to give the pattern; in layer 6 all the 80 are replaced with 1 to give the pattern.

Using the weighted average, we get a level of agreement of 0.61 in this example. The outlined procedure can be applied to any ordered frequency distribution.

3.2 Interpretation of Agreement Scores

Levels of agreement range from -1 to 1. There are three ideal-typical positions that help interpretation of the scores. If all respondents agree on the category (position), agreement is 1. This is illustrated in the top-left corner of the figure. If the respondents are evenly spread, and each category has the same number of responses, agreement is 0. This is illustrated at the top right of the figure. If respondents are divided, and half the responses are in one extreme category, and the other half are in the other extreme category, agreement is -1. This is shown at the bottom left of the figure. The following figure also includes an example between these ideal types (bottom right). Agreement is only defined if there are at least three response categories, and it does not tell you which of the categories is the most common one. For this reason, it is also advisable to look at an appropriate measure of central tendency, such as the interpolated median.





3.3 Calculating Agreement

To calculate agreement, we simply call `agreement` with the frequency vector as the argument. There is normally *no reason* to use the old algorithm, which can be set using `old=TRUE`. This is the 'old' algorithm as described by Van der Eijk.

Let us illustrate this using the data from above.

```
> x = c(30, 40, 210, 130, 530, 50, 10) # these are our data
> agreement(x)

[1] 0.6113333
```

Here are two additional examples, using data from van der Eijk's article. Van der Eijk gives the example of respondents placing political parties on a 7-point rating scale.

Position	1	2	3	4	5	6	7
PvdA	2.4%	2.8%	3.2%	6.2%	13.5%	30.4%	41.6%
D66	1.6%	2.6%	8.2%	21%	29.3%	27%	10.3%

To calculate the level of agreement for these two frequency distributions, we can simply type `agreement(c(2.4,2.8,3.2,6.2,13.5,30.4,41.6))` for the PvdA, and `agreement(c(1.6,2.6,8.2,21,29.3,27,10.3))` for the D66. This gives us levels of agreement of 0.61 for the PvdA, and 0.48 for the D66.

```
> agreement(c(2.4,2.8,3.2,6.2,13.5,30.4,41.6)) # PvdA

[1] 0.6097236

> agreement(c(1.6,2.6,8.2,21,29.3,27,10.3))    # D66

[1] 0.4823333
```

3.4 Using Agreement with Survey Data

When using survey data (or other similarly structured data), it is easy to call `agreement` together with the helper function and a variable name (e.g. var99): `agreement(collapse(var99, pos=0:7))`. Subscripting works as expected: `agreement(collapse(var99[country=="UK"], pos=0:7))`, so `sapply` plays nicely.

3.5 Agreement A with and without Simulated Coding Error

When agreement is calculated on ordered rating scales from sources that are manually coded, the influence of coding error on the resulting value of agreement can result in uncertainty. Using simulated coding error, it is possible to express this uncertainty, subject to certain assumptions.

The function `compareAgreement` returns a list with agreement 'A' without simulated coding errors (i.e. simply calculated), the mean of agreement with simulated coding error, and the standard deviation of agreement with simulated coding error. Default values for the simulation are: n=500 samples, e=0.01 proportion of samples that are simulated as mis-coded, N=500 replications for calculating the mean and standard deviation. Given that these coding errors are simulated, each run will give a different value of the mean and standard deviation.

The helper function `agreementError` is used for the actual simulations, so different measures of central tendency can be calculated. Contrary to bootstrapping, it is not necessary to have all positions observed in the frequency vector. To understand the advantage, take an extreme example: [30000]. Here we have three observations at the first position, but none at the others. Bootstrapping will always lead to the same agreement score (perfect agreement). 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 of these answers could refer to 'somewhat agree'. The 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.

Usage: `compareAgreement(expand(V))`.

Usage: `compareAgreement(c(1,1,1), pos=1:5)`

3.6 Calculating Polarization

The agreement scores are suitable to express polarization, but the numbers are not intuitive to interpret. The function `polarization` offers easily interpretable values by re-scaling agreement A. More precisely, it gives $(1 - agreement)/2$. This means that a polarization value of 1 means perfect polarization (bottom-left corner of the figure above), and a value of 0 means perfect agreement. A value of 0.5 corresponds to the 'no agreement' in the above graph.

Usage is equivalent to `agreement`. In the first example, we specify the entire data, use the `collapse` helper function to determine the frequency vector (note: there are no observations at position 6, even though this position is permitted), and calculate polarization values.

```
> polarization(collapse(c(1,2,4,2,5,2,7,7,3,1,2,1,3,2,4,
+ 1,5,2,3,2,4,2,3,1,1,3), pos=1:7))

[1] 0.3038462
```

Or to continue using the Dutch parties:

```
> polarization(c(2.4,2.8,3.2,6.2,13.5,30.4,41.6)) # PvdA

[1] 0.1951382

> polarization(c(1.6,2.6,8.2,21,29.3,27,10.3))    # D66

[1] 0.2588333
```

In a typical setting, we would probably use a variable name from a dataset (e.g. POSITION): `polarization(collapse(POSITION, pos=0:5))`.

4 Leik's Consensus

Leik [1966] introduced a measure of ordinal dispersion in 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. The van der Eijk measure has a fixed mid-point that is independent of the number of categories. 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.

Leik's ordinal 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.

5 Tatsle and Wierman's Consensus

Tastle and Wierman [2007] take a different approach to measuring consensus, using the Shannon entropy as the basis.

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.

Usage: `consensus(V)`.

5.1 Entropy

The Shannon entropy can be calculated using `entropy`. This function follows Tastle and Wierman [2007] and ignores categories with zero observations. The Shannon entropy ignores the order of categories; use `consensus` to consider the order of categories.

6 Berry and Mielke's IOV

Berry and Mielke [1992] introduced a measure of dispersion based on squared Euclidean distances (IOV). Contrary to the presentation in Blair and Lacy [2000], the adjustment for Tmax omitted by Blair and Lacy is included here, as there is no reason to leave it out when a computer does the calculation. A derived measure `COV` by Kvalseth [1995] is implemented separately as `Kvalseth`. Usually, the IOV is equivalent to 1-`lsquared` by Blair and Lacy [2000].

6.1 Kvalseth COV

`Kvalseth` COV is a measure of dispersion derived from `BerryMielke`, based on Euclidean distances.

7 Blair and Lacy's "l" (squared)

Blair and Lacy [2000] introduced a measure of concentration "l" (squared), based on linear Euclidean distances. It is implemented as `lsquared`.

7.1 "d" (squared)

Blair and Lacy [2000] introduced a measure of concentration "d" squared, based on linear Euclidean distances. This measure does not normalize values, which is done by `lsquared`. The non-normalized measure is implemented as `dsquared`.

8 Garcia-Montalvo and Reynal-Querol

Garcia-Montalvo and Reynal-Querol [2005] calculates polarization, as introduced by Reynal-Querol [2002]. This measure is known as the Reynal-Querol index of polarization (RQ), Montalvo and Reynal-Querol (MRQ), or Garcia-Montalvo and Reynal-Querol. The 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.

9 AJUS

The package includes a few additional functions for frequency vectors. The AJUS classification of distributions is one: Galtung [1969] introduced a system to classify distributions according to shape. This is a means to reduce complexity. For full details, refer to Galtung's book on social research.

`ajus(distribution)` gives you the shape or type of the distribution, as well as whether there is a skew. I have added two new types F and L to complement the ones identified by Galtung. The skew is given as -1 for a negative skew, 0 for absence of skew, or $+1$ for a positive skew.

You can choose whether to use a strict AJUS system following Galtung, or use the modified AJUSFL system that includes the L and F types. The default is the modified variant. The modified system draws a distinction 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 strict AJUS system has no F type and returns NA instead.

- A: unimodal distribution, peak in the middle
- J: unimodal, peak at either end (strict) or peak on right (modified)
- L: unimodal, peak on left (modified only)

- U: bimodal, peak at both ends
- S: bimodal or multi-modal, multiple peaks
- F: flat, no peak (modified only)

Galtung developed the AJUS system for a somewhat systematic classification of distributions, but not for the use on computers. The advantage of using a function on the computer is twofold. On the one hand, we can easily apply the AJUS system to many distributions, `sapply` may be your friend there. On the other hand, the *tolerance* used in the AJUS system is applied systematically. When using human judgement on whether two values are 'roughly the same' or different, a really systematic approach cannot be guaranteed. In the AJUS function, you can specify the argument `tolerance` to change the tolerance. The AJUS function ignores all differences equal to or smaller than the tolerance parameter. The package default is 0.1, possibly useful when working with values between 0 and 1, in which case it corresponds to 10 per cent. The `tolerance` parameter is not a trivial choice, and it can affect results.

```
> # Example 1: different types of distributions
> V1 = c(0,1,0)     # A
> ajus(V1)

$type
[1] "A"

$skew
[1] 0

$pattern
[1]  1 -1

> V2 = c(0,0,1)     # J
> ajus(V2)

$type
[1] "J"

$skew
[1] 1

$pattern
[1] 0 1

> V3 = c(1,0,1)     # U
> ajus(V3)
```

```
$type
[1] "U"

$skew
[1] 0

$pattern
[1] -1  1

> V4 = c(1,0,1,0)   # S
> ajus(V4)

$type
[1] "S"

$skew
[1] 0

$pattern
[1] -1  1 -1

> V5 = c(0,0,0)     # F
> ajus(V5)

$type
[1] "F"

$skew
[1] 0

$pattern
[1] 0 0

> V6 = c(1,0,0)     # L
> ajus(V6)

$type
[1] "L"

$skew
[1] -1

$pattern
[1] -1  0

> ajus(V6, variant="strict") # gives J
```

```
$type
[1] "J"

$skew
[1] -1

$pattern
[1] -1  0
```

Additional functions are provided to use the AJUS system: `ajusCheck` takes a vector of tolerance values to test the sensitivity of the classification to the tolerance parameter; `ajusPlot` plots the distribution along with its type and skew. The latter function can deal with missing values, such as when using time series data with missing values.

```
> # Example 2: varying tolerance to check sensitivity
> V7 = c(0,0,1,2,1)
> ajus(V7, tolerance=0.5)

$type
[1] "A"

$skew
[1] 1

$pattern
[1]  0  1  1 -1

> ajus(V7, tolerance=1)

$type
[1] "F"

$skew
[1] 1

$pattern
[1] 0 0 0 0

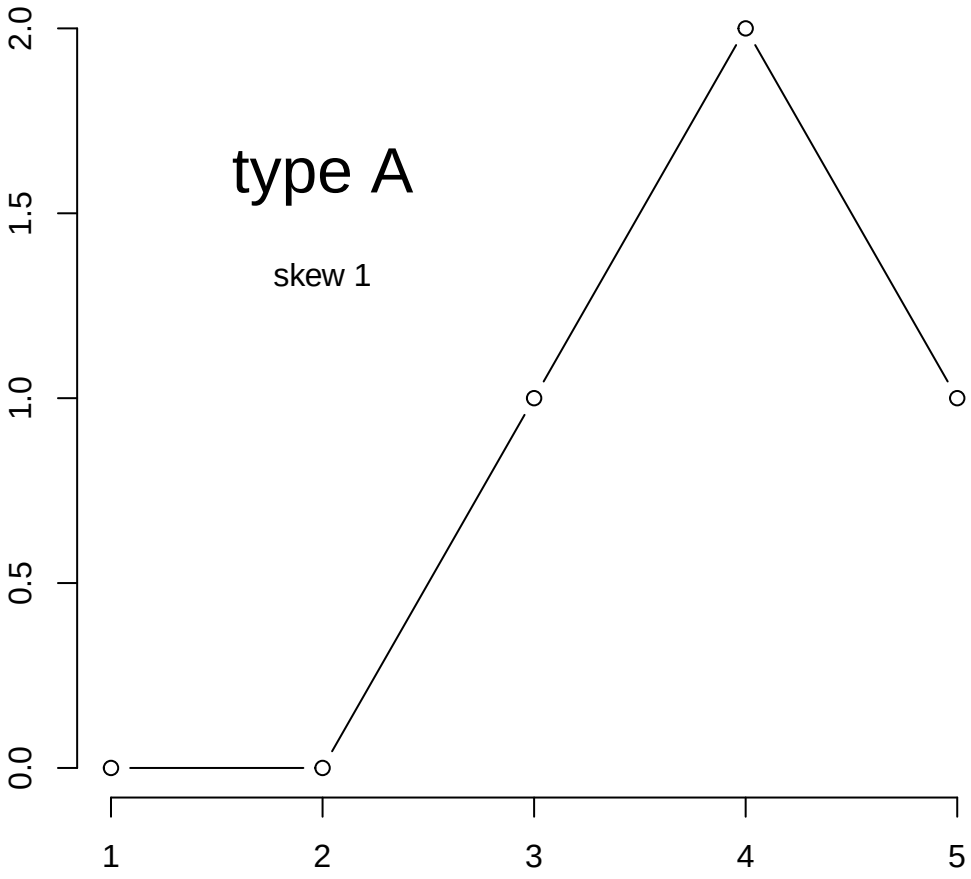
> ajusCheck(V7, t=c(0.1, 0.5, 0.6, 1))

$tolerance
[1] 0.1 0.5 0.6 1.0

$type
[1] "A" "A" "A" "F"

$skew
[1] 1 1 1 1
```

```
> # Example 3: plotting AJUS
> ajusPlot(V7)
```



10 ISD

The ISD system by Galtung [1969] is another way to reduce complexity, this time for changes over time. The ISD takes a vector with three time points. These three points describe two periods during which changes may occur (that is, two

transition points). `isd(distribution)` gives you a type (i.e. the number) and a description of the type (i.e. the description in words).

- Type 1: increase in both periods
- Type 2: increase in first period, flat in second period
- Type 3: increase in first period, decrease in second period
- Type 4: flat in first period, increase in second period
- Type 5: flat in both periods
- Type 6: flat in first period, decrease in second period
- Type 7: decrease in first period, increase in second period
- Type 8: decrease in first period, flat in second period
- Type 9: decrease in both periods

The tolerance parameter works like described in the previous section on AJUS.

11 (Multiple) Modes

The function `modes` tells you at which position the mode is. This can be used, for instance, jointly with the agreement function to identify at which point agreement is reached (not only that). The function accepts frequency distributions where multiple positions are the most common ones, which can happen in ordered rating scales. The function `secondModes` additionally gives you the value and position(s) of the second most common value. In addition to the mode and the positions, the functions also indicate whether these values are contiguous (i.e. in neighbouring response categories).

A `tolerance` parameter is used to ignore small differences. See the section on AJUS above for details.

Let us look at made-up examples to illustrate the function. The output first shows the categories at which observations occur; use the parameter `pos` to specify the values. If there are categories with 0 observations, these need to be indicated in the frequency vector accordingly (you might want to use the helper function `collapse`. 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 discarded. Second, it shows the number of observations at each position. This is the frequency vector. The mode is identified, and the position at which it occurs. If there are multiple positions where the mode occurs, these are all listed. Finally, it is stated whether the modes are contiguous (true/false). This factor is interesting if there are multiple modes – if they are not contiguous, this can be understood as polarization; with a single mode it is always 'contiguous'.

In the first example, we look at a simple frequency vector. No categories are specified (`pos` argument not declared), so the categories are assumed to be 1:7 in this case.

```
> # Example 1: finding the mode
> V1 = c(30,40,210,130,530,50,10)
> modes(V1) # will find position 5

$at
[1] 1 2 3 4 5 6 7

$frequencies
[1]  30  40 210 130 530  50  10

$mode
[1] 5

$positions
[1] 5

$contiguous
[1] TRUE
```

Here is a shorter frequency vector, again without specifying the categories — 1 to 4 are assumed.

```
> # Example 2:
> V2 = c(3,0,4,1)
> modes(V2) # will find position 3

$at
[1] 1 2 3 4

$frequencies
[1] 3 0 4 1

$mode
[1] 3

$positions
[1] 3

$contiguous
[1] TRUE
```

This example illustrates the `pos` argument. The frequency vector from the previous example is reused, but it is specified that the categories refer to values of $[-1, 0, 1, 2]$. The mode identified reflects this change, while the position is unchanged (since the same frequency vector is used).

```
> # Example 3: providing categories
> modes(V2,pos=-1:2) # will still find position 3, but give the value of 1 as mode

$at
[1] -1  0  1  2

$frequencies
[1] 3 0 4 1

$mode
[1] 1

$positions
[1] 3

$contiguous
[1] TRUE
```

This example illustrates multiple modes. The frequencies of 528 and 530 are nearly the same, that is to say given the (default) tolerance they are considered the same.

```
> # Example 4: similar values
> V3 = c(30,40,510,130,530,50,10)
> modes(V3, tolerance=30) # will find positions 3 and 5 (510 and 530 are nearly the same)

$at
[1] 1 2 3 4 5 6 7

$frequencies
[1]  30  40 510 130 530  50  10

$mode
[1] 3 5

$positions
[1] 3 5

$contiguous
[1] FALSE
```

12 Going Further

For further documentation, please refer to the original contributions by Van der Eijk [2001], Leik [1966], Tastle and Wierman [2007], Berry and Mielke [1992], Blair and Lacy [2000], Reynal-Querol [2002], and Galtung [1969], and refer to the package help files.

For more details on the different measures of agreement and how they compare, see: Aeppli, Clem, and Didier Ruedin. 2026. Let's Measure Concentration and Dispersion in Ordinal Data. Methods, Data, Analyses. Forthcoming.

References

K. Berry and P. Mielke. Assessment of variation in ordinal data. *Perceptual and Motor Skills*, 74(1):63–66, 1992. doi: 10.2466/pms.1992.74.1.63.

J. Blair and M. Lacy. Statistics of Ordinal Variation. *Sociological Methods & Research*, 28(3):251–280, 2000. doi: 10.1177/0049124100028003001.

J. Galtung. *Theory and Methods of Social Research*. Universitetsforlaget, Oslo, 1969. ISBN 0043000177.

Jose G. Garcia-Montalvo and Marta Reynal-Querol. Ethnic diversity and economic development. *Journal of Development Economics*, 76(2):293–323, 2005.

T. Kvalseth. Coefficients of variation for nominal and ordinal categorical data. *Perceptual and Motor Skills*, 80(3):843–847, 1995. doi: 10.2466/pms.1995.80.3.843.

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

Marta Reynal-Querol. Ethnicity, political systems, and civil wars. *Journal of Conflict Resolution*, 46(1):29–54, 2002.

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

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