Package ‘nomordR’

May 20, 2021

Title Randomization Test for Sequences of Nominal Values

Version 0.1

Description Implements the nomord_probe() function, which performs a statistical analysis on an input vector (sequence) of nominal (categorical) values.

Imports methods

License GPL-3

Encoding UTF-8

LazyData false

RoxygenNote 7.1.1

NeedsCompilation no

Author Peter Szabo [aut, cre],
Andras Szilagyi [aut],
Janos Podani [aut]

Maintainer Peter Szabo <peterszabomail77@gmail.com>

Repository CRAN

Date/Publication 2021-05-20 07:00:06 UTC

R topics documented:

`nomordR` *nomordR: Implementation of a Randomization Test for Sequences of Nominal Values*

Description

The included nomord_probe function performs a statistical analysis on an input vector (sequence) of nominal (categorical) values. The vector positions and the corresponding nominal values define ordinal-nominal value pairs, which can be associated. The function first calculates an association statistic, then performs a randomization test to check for a significant association.

Usage

The nomord_probe function has two mandatory arguments; the input sequence as a vector (obligatory) and the chosen statistic ('U' or 'T'). The command returns an S4 object with slot names input, statname, statvalue, althyp and p, standing for the input sequence, the chosen statistic, the (normalized) statistic value, the textual alternative hypothesis, and the p-value of the statistical test, respectively. Printing the result also displays these values. A statistic ('U' or 'T') value of –1 indicates maximum segregation, while a value of 1 indicates maximum aggregation of the nominal values along the ordinal dimension.

Examples:

```
> v = c("B","A","A","C","C","B","A","A")
> nomord_probe(v, "U")
Association test for a sequence of nominal values
input:  B A A C C B A A
U -statistic:  -0.4285714
alternative hypothesis: U statistic is smaller than expected from a random sequence
p-value:  0.2723
>
> result = nomord_probe(v, "U")
> result@p
[1] 0.2723
```

Computational background

Two association metrics are available. The U statistic expresses the topological segregation of the sequence by counting the number of other types appearing between every pair of vector elements, denoted by w_{ij}. The unnormalized U' is calculated as $U' = \sum_{i<j} w_{ij}$, and the normalized U value is calculated as $U = (U' - U_{min})/(U_{max} - U_{min})$, where U'_{max} and U'_{min} are the theoretical largest and smallest U' values for a sequence with the same length and the same nominal value distribution.

The T statistic expresses the subsequence diversity by counting the number of different elements in all possible v-long substrings. The window width v is equal to the number of distinct nominal values. Denoting the number of different elements in a substring starting at index i as d_i, the unnormalized T' is calculated as $U' = \sum_i d_i$, and the normalized T value is calculated as $T = (T' - T_{min})/(T_{max} - T_{min})$, where T'_{max} and T'_{min} are the theoretical largest and smallest T' values for a sequence with the same length and the same nominal value distribution.

U_{min} is equal to zero. The other extremum values are calculated from algorithmically generated sequences, with the same length (L) and nominal-value occurrences ($n_c, c = \{1, 2, ..., k\}$) as the input vector. (The validity of these algoriths was checked with a Metropolis–Hastings algorithm.)

U_{max}:

1. Assign *p* priority values to all elements $(i = \{1, 2, ..., n_c\})$ of all types $(c = \{1, 2, ..., k\})$ according to the rule $p = i/(n_c + 1)$, where k is the number of different nominal values.

2. Order the elements according to their priorities. Break ties between elements of different types consistently according to any arbitrary ordering of types.

———-

T_{min}

Create a sequence consisting of homogeneous blocks of the different elements. Place shorter blocks (with smaller n_c) as close to either the left or right ends, as possible.

———-

T_{max}

1. Create an indexed list of the different types (e.g. [1, 2, 3, 4]). Initially, the number of available elements for all types (m_c) is equal to their occurrence values (n_c).

2. Assign elements to the different places of the sequence, starting from the middle and going towards the left and right ends, using an auxiliary variable i ranging from 1 to L (sequence length). For each value of i:

(a) The place to assign a value to is; *p = floor(L/2)*, if i is odd, *p = floor(L/2) + i/2*, if i is even.

(b) Let the candidate to this place be the type with the index *p mod k*, denoted by c.

(c) If m_c is zero, then let c be the type with the highest remaining number (m_c).

(d) Assign c to position *p* of the sequence and decrease m_c with one.

This algorithm fills a sequence with elements starting from the middle and going towards the left and right ends. It tries to place different types next to each other, but if it is not possible (there are no more elements of the desired type), then it is replaced by an element of the most abundant type.

`nomord_probe` *nomord_probe*

Description

Returns the probability p

Usage

```
nomord_probe(seq, statname)
```

Arguments

`seq`	A vector of categorical variables
`statname`	The used statistic function ("U" or "T")

Value

A nomord class S4 object with the following slots: input, statname, statvalue, althyp, p

Examples

```
nomord_probe(c("B","A","A","C","C","B","A","A"), 'U')
```

Index