# Package ‘dixonTest’

October 13, 2022

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Dixon's Ratio Test for Outlier Detection</td>
</tr>
<tr>
<td>Version</td>
<td>1.0.4</td>
</tr>
<tr>
<td>Date</td>
<td>2022-08-22</td>
</tr>
</tbody>
</table>

**Description**  For outlier detection in small and normally distributed samples the ratio test of Dixon (Q-test) can be used. Density, distribution function, quantile function and random generation for Dixon's ratio statistics are provided as wrapper functions. The core applies McBane's Fortran functions &lt;[doi:10.18637/jss.v016.i03]&gt; that use Gaussian quadrature for a numerical solution.

<table>
<thead>
<tr>
<th>License</th>
<th>GPL-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ByteCompile</td>
<td>yes</td>
</tr>
<tr>
<td>NeedsCompilation</td>
<td>yes</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>Classification/MSC-2010</td>
<td>62F03, 62E17, 62Q05</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>7.2.0</td>
</tr>
</tbody>
</table>

**Author** Thorsten Pohlert [aut, cre] (&lt;https://orcid.org/0000-0003-3855-3025&gt;), George C. McBane [ctb]

**Maintainer** Thorsten Pohlert &lt;thorsten.pohlert@gmx.de&gt;

**Repository** CRAN

**Date/Publication** 2022-08-22 19:40:02 UTC

## R topics documented:

- Dixon
- dixonTest

**Index**

<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
Description

Density, distribution function, quantile function and random generation for Dixon’s ratio statistics 
\( r_{j,i-1} \) for outlier detection.

Usage

qdixon(p, n, i = 1, j = 1, log.p = FALSE, lower.tail = TRUE)
pdixon(q, n, i = 1, j = 1, lower.tail = TRUE, log.p = FALSE)
ddixon(x, n, i = 1, j = 1, log = FALSE)
rdixon(n, i = 1, j = 1)

Arguments

- \( p \) vector of probabilities.
- \( n \) number of observations. If length(\( n \)) > 1, the length is taken to be the number required
- \( i \) number of observations <= \( x_i \)
- \( j \) number of observations >= \( x_j \)
- \( \log.p \) logical; if TRUE probabilities \( p \) are given as \( \log(p) \)
- \( \text{lower.tail} \) logical; if TRUE (default), probabilities are \( P[X <= x] \) otherwise, \( P[X > x] \).
- \( q \) vector of quantiles
- \( x \) vector of quantiles.
- \( \log \) logical; if TRUE (default), probabilities \( p \) are given as \( \log(p) \).

Details

According to McBane (2006) the density of the statistics \( r_{j,i-1} \) of Dixon can be yield if \( x \) and \( v \) are integrated over the range \( (-\infty < x < \infty, 0 \leq v < \infty) \)

\[
f(r) = \frac{n!}{(i-1)!(n-j-i-1)!(j-1)!} \times \int_{-\infty}^{\infty} \int_{0}^{\infty} \phi(t)dt \left[ \int_{-\infty}^{x-v} \phi(t)dt \right]^{i-1} \left[ \int_{x-v}^{x-rv} \phi(t)dt \right]^{n-j-i-1} \times \left[ \int_{x-rv}^{x-v} \phi(t)dt \right]^{j-1} \phi(x-v)\phi(x-rv)\phi(x)v dv dx
\]

where \( v \) is the Jacobian and \( \phi(.) \) is the density of the standard normal distribution. McBane (2006) has proposed a numerical solution using Gaussian quadratures (Gauss-Hermite quadrature and half-range Hermite quadrature) and coded a library in Fortran. These R functions are wrapper functions to use the respective Fortran code.
Value

dixon gives the density function, pdixon gives the distribution function, qdixon gives the quantile function and rdixon generates random deviates.

Source

The R code is a wrapper to the Fortran code released under GPL >=2 in the electronic supplement of McBane (2006). The original files are 'rfuncs.f', 'utility.f' and 'dixonr.fi'. They were slightly modified to comply with current CRAN policy and the R manual 'Writing R Extensions'.

Note

The file 'slowTest/d-p-q-r-tests.R.out.save' that is included in this package contains some results for the assessment of the numerical accuracy.

The slight numerical differences between McBane’s original Fortran output (see files 'slowTests/test[1,2,4].ref.output.txt') and this implementation are related to different floating point rounding algorithms between R (see ‘round to even’ in round) and Fortran’s write(*, 'F6.3') statement.

References


Examples

```r
set.seed(123)
n <- 20
Rdixon <- rdixon(n, i = 3, j = 2)
Rdixon
pdixon(Rdixon, n = n, i = 3, j = 2)
ddixon(Rdixon, n = n, i = 3, j = 2)
```

dixonTest

Dixons Outlier Test (Q-Test)

Description

Performs Dixons single outlier test.

Usage

dixonTest(x, alternative = c("two.sided", "greater", "less"), refined = FALSE)
dixonTest

Arguments

- **x**: a numeric vector of data
- **alternative**: the alternative hypothesis. Defaults to "two.sided"
- **refined**: logical indicator, whether the refined version or the Q-test shall be performed. Defaults to FALSE

Details

Let $X$ denote an identically and independently distributed normal variate. Further, let the increasingly ordered realizations denote $x_1 \leq x_2 \leq \ldots \leq x_n$. Dixon (1950) proposed the following ratio statistic to detect an outlier (two sided):

$$r_{j,i-1} = \max \left\{ \frac{x_n - x_{n-j}}{x_n - x_i}, \frac{x_{1+j} - x_1}{x_{n-i} - x_1} \right\}$$

The null hypothesis, no outlier, is tested against the alternative, at least one observation is an outlier (two sided). The subscript $j$ on the $r$ symbol indicates the number of outliers that are suspected at the upper end of the data set, and the subscript $i$ indicates the number of outliers suspected at the lower end. For $r_{10}$ it is also common to use the statistic $Q$.

The statistic for a single maximum outlier is:

$$r_{j,i-1} = \frac{(x_n - x_{n-j})}{(x_n - x_i)}$$

The null hypothesis is tested against the alternative, the maximum observation is an outlier.

For testing a single minimum outlier, the test statistic is:

$$r_{j,i-1} = \frac{(x_{1+j} - x_1)}{(x_{n-i} - x_1)}$$

The null hypothesis is tested against the alternative, the minimum observation is an outlier.

Apart from the earlier Dixons Q-test (i.e. $r_{10}$), a refined version that was later proposed by Dixon can be performed with this function, where the statistic $r_{j,i-1}$ depends on the sample size as follows:

- $r_{10}$: $3 \leq n \leq 7$
- $r_{11}$: $8 \leq n \leq 10$
- $r_{21}$: $11 \leq n \leq 13$
- $r_{22}$: $14 \leq n \leq 30$

The p-value is computed with the function `pdixon`.

References


Examples

```r
x <- c(40.02, 40.12, 40.16, 40.18, 40.18, 40.20)
dixonTest(x, alternative = "two.sided")

## example from the dataplot manual of NIST
x <- c(568, 570, 570, 570, 572, 578, 584, 596)
dixonTest(x, alternative = "greater", refined = TRUE)
```
Index

* distribution
  Dixon, 2
* htest
  dixonTest, 3
* outliers
  dixonTest, 3

ddixon (Dixon), 2
Dixon, 2
dixonTest, 3

pdxion, 4
pdxion (Dixon), 2

qdixon (Dixon), 2
rdixon (Dixon), 2
round, 3