

# Package: jaccard (via r-universe)

September 6, 2024

**Type** Package

**Title** Testing similarity between binary datasets using  
Jaccard/Tanimoto coefficients

**Version** 0.1.0

**Date** 2017-09-04

**Author** Neo Christopher Chung <nchchung@gmail.com>, Błażej Miasojedow  
<btriasojedow@gmail.com>, Michał Startek  
<M.Startek@mimuw.edu.pl>

**Maintainer** Neo Christopher Chung <nchchung@gmail.com>

**Description** Calculate statistical significance of Jaccard/Tanimoto  
similarity coefficients.

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**Imports** Rcpp (>= 0.12.6), qvalue, dplyr, magrittr, shiny

**LinkingTo** Rcpp

**NeedsCompilation** yes

**SystemRequirements** C++11

**RoxygenNote** 6.0.1

**Repository** <https://ncchung.r-universe.dev>

**RemoteUrl** <https://github.com/ncchung/jaccard>

**RemoteRef** HEAD

**RemoteSha** ad1d8fc43ad270a6dea1d489fdcde8dc84fa7ec6

## Contents

jaccard . . . . .	2
jaccard.ev . . . . .	3
jaccard.rahman . . . . .	3

jaccard.test . . . . .	4
jaccard.test.asymptotic . . . . .	6
jaccard.test.bootstrap . . . . .	7
jaccard.test.exact . . . . .	8
jaccard.test.mca . . . . .	8
jaccard.test.pairwise . . . . .	9
runJaccardApp . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

jaccard	<i>Compute a Jaccard/Tanimoto similarity coefficient</i>
---------	--

---

## Description

Compute a Jaccard/Tanimoto similarity coefficient

## Usage

```
jaccard(x, y, center = FALSE, px = NULL, py = NULL)
```

## Arguments

<code>x</code>	a binary vector (e.g., fingerprint)
<code>y</code>	a binary vector (e.g., fingerprint)
<code>center</code>	whether to center the Jaccard/Tanimoto coefficient by its expectation
<code>px</code>	probability of successes in <code>x</code> (optional)
<code>py</code>	probability of successes in <code>y</code> (optional)

## Value

`jaccard.test.bootstrap` returns an expected value.

## Examples

```
set.seed(1234)
x = rbinom(100,1,.5)
y = rbinom(100,1,.5)
jaccard(x,y)
```

---

<code>jaccard.ev</code>	<i>Compute an expected Jaccard/Tanimoto similarity coefficient under independence</i>
-------------------------	---

---

**Description**

Compute an expected Jaccard/Tanimoto similarity coefficient under independence

**Usage**

```
jaccard.ev(x, y, px = NULL, py = NULL)
```

**Arguments**

<code>x</code>	a binary vector (e.g., fingerprint)
<code>y</code>	a binary vector (e.g., fingerprint)
<code>px</code>	probability of successes in <code>x</code> (optional)
<code>py</code>	probability of successes in <code>y</code> (optional)

**Value**

`jaccard.test.bootstrap` returns an expected value.

**Examples**

```
set.seed(1234)
x = rbinom(100,1,.5)
y = rbinom(100,1,.5)
jaccard.ev(x,y)
```

---

<code>jaccard.rahman</code>	<i>Compute p-value using the EC-BLAST method</i>
-----------------------------	--

---

**Description**

In the EC-BLAST paper, Rahman et al. (2014) provide the following description: The mean ( $\mu$ ) and s.d. ( $\sigma$ ) of the similarity scores are used to define the z score,  $z = (T_w - \mu) / \sigma$ . For the purpose of calculating the P value, only hits with  $T > 0$  are considered. The P value  $w$  is derived from the z score using an extreme value distribution  $P = 1 - \exp(-e^{-z} / \sqrt[6]{6} - \Gamma(1))$ , where the Euler-Mascheroni constant  $\Gamma(1) = 0.577215665$ .

**Usage**

```
jaccard.rahman(j)
```

**Arguments**

`j` a numeric vector of observed Jaccard coefficients (uncentered)

**Value**

`jaccard.rahman` returns a numeric vector of p-values

**References**

Rahman, Cuesta, Furnham, Holliday, and Thornton (2014) EC-BLAST: a tool to automatically search and compare enzyme reactions. *Nature Methods*, 11(2) <http://www.nature.com/nmeth/journal/v11/n2/full/nmeth.2803.html>

---

<code>jaccard.test</code>	<i>Test for Jaccard/Tanimoto similarity coefficients</i>
---------------------------	--

---

**Description**

Compute statistical significance of Jaccard/Tanimoto similarity coefficients between binary vectors, using four different methods.

**Usage**

```
jaccard.test(x, y, method = "mca", px = NULL, py = NULL, verbose = TRUE,
...)
```

**Arguments**

`x` a binary vector (e.g., fingerprint)  
`y` a binary vector (e.g., fingerprint)  
`method` a method to compute a p-value ("mca", "bootstrap", "asymptotic", or "exact")  
`px` probability of successes in `x` (optional)  
`py` probability of successes in `y` (optional)  
`verbose` whether to print progress messages  
`...` optional arguments for specific computational methods

**Details**

There exist four methods to compute p-values of Jaccard/Tanimoto similarity coefficients: `mca`, `bootstrap`, `asymptotic`, and `exact`. This is simply a wrapper function for corresponding four functions in this package: `jaccard.test.mca`, `jaccard.test.bootstrap`, `jaccard.test.asymptotic`, and `jaccard.test.exact`.

We recommend using either `mca` or `bootstrap` methods, since the `exact` solution is slow for a moderately large vector and `asymptotic` approximation may be inaccurate depending

on the input vector size. The bootstrap method uses resampling with replacement binary vectors to compute a p-value (see optional arguments). The `mca` method uses the measure concentration algorithm that estimates the multinomial distribution with a known error bound (specified by an optional argument `accuracy`).

### Value

`jaccard.test` returns a list mainly consisting of

<code>statistics</code>	centered Jaccard/Tanimoto similarity coefficient
<code>pvalue</code>	p-value
<code>expectation</code>	expectation

### Optional arguments for `method="bootstrap"`

**fix** whether to fix (i.e., not resample) `x` and/or `y`

**B** a total bootstrap iteration

**seed** a seed for a random number generator

### Optional arguments for `method="mca"`

**accuracy** an error bound on approximating a multinomial distribution

**error.type** an error type on approximating a multinomial distribution ("`average`", "`upper`", "`lower`")

**seed** a seed for the random number generator.

### See Also

[jaccard.test.bootstrap](#) [jaccard.test.mca](#) [jaccard.test.exact](#) [jaccard.test.asymptotic](#)

### Examples

```
set.seed(1234)
x = rbinom(100,1,.5)
y = rbinom(100,1,.5)
jaccard.test(x,y,method="bootstrap")
jaccard.test(x,y,method="mca")
jaccard.test(x,y,method="exact")
jaccard.test(x,y,method="asymptotic")
```

---

`jaccard.test.asymptotic`*Compute p-value using an asymptotic approximation*

---

## Description

Compute statistical significance of Jaccard/Tanimoto similarity coefficients.

## Usage

```
jaccard.test.asymptotic(x, y, px = NULL, py = NULL, verbose = TRUE)
```

## Arguments

<code>x</code>	a binary vector (e.g., fingerprint)
<code>y</code>	a binary vector (e.g., fingerprint)
<code>px</code>	probability of successes in <code>x</code> (optional)
<code>py</code>	probability of successes in <code>y</code> (optional)
<code>verbose</code>	whether to print progress messages

## Value

`jaccard.test.asymptotic` returns a list consisting of

<code>statistics</code>	centered Jaccard/Tanimoto similarity coefficient
<code>pvalue</code>	p-value
<code>expectation</code>	expectation

## Examples

```
set.seed(1234)
x = rbinom(100,1,.5)
y = rbinom(100,1,.5)
jaccard.test.asymptotic(x,y)
```

---

`jaccard.test.bootstrap`*Compute p-value using the bootstrap procedure*

---

## Description

Compute statistical significance of Jaccard/Tanimoto similarity coefficients.

## Usage

```
jaccard.test.bootstrap(x, y, px = NULL, py = NULL, verbose = TRUE,  
  fix = "x", B = 1000, seed = NULL)
```

## Arguments

<code>x</code>	a binary vector (e.g., fingerprint)
<code>y</code>	a binary vector (e.g., fingerprint)
<code>px</code>	probability of successes in x (optional)
<code>py</code>	probability of successes in y (optional)
<code>verbose</code>	whether to print progress messages
<code>fix</code>	whether to fix (i.e., not resample) x and/or y
<code>B</code>	a total bootstrap iteration
<code>seed</code>	a seed for a random number generator

## Value

`jaccard.test.bootstrap` returns a list consisting of

<code>statistics</code>	centered Jaccard/Tanimoto similarity coefficient
<code>pvalue</code>	p-value
<code>expectation</code>	expectation

## Examples

```
set.seed(1234)  
x = rbinom(100,1,.5)  
y = rbinom(100,1,.5)  
jaccard.test.bootstrap(x,y,B=500)
```

---

`jaccard.test.exact`     *Compute p-value using the exact solution*

---

### Description

Compute statistical significance of Jaccard/Tanimoto similarity coefficients.

### Usage

```
jaccard.test.exact(x, y, px = NULL, py = NULL, verbose = TRUE)
```

### Arguments

<code>x</code>	a binary vector (e.g., fingerprint)
<code>y</code>	a binary vector (e.g., fingerprint)
<code>px</code>	probability of successes in <code>x</code> (optional)
<code>py</code>	probability of successes in <code>y</code> (optional)
<code>verbose</code>	whether to print progress messages

### Value

`jaccard.test.exact` returns a list consisting of

<code>statistics</code>	centered Jaccard/Tanimoto similarity coefficient
<code>pvalue</code>	p-value
<code>expectation</code>	expectation

### Examples

```
set.seed(1234)
x = rbinom(100,1,.5)
y = rbinom(100,1,.5)
jaccard.test.exact(x,y)
```

---

`jaccard.test.mca`     *Compute p-value using the Measure Concentration Algorithm*

---

### Description

Compute statistical significance of Jaccard/Tanimoto similarity coefficients.

### Usage

```
jaccard.test.mca(x, y, px = NULL, py = NULL, accuracy = 1e-05,
  error.type = "average", verbose = TRUE)
```



**Arguments**

<code>x</code>	a binary vector (e.g., fingerprint)
<code>y</code>	a binary vector (e.g., fingerprint)
<code>px</code>	probability of successes in <code>x</code> (optional)
<code>py</code>	probability of successes in <code>y</code> (optional)
<code>accuracy</code>	an error bound on approximating a multinomial distribution
<code>error.type</code>	an error type on approximating a multinomial distribution ("average", "upper", "lower")
<code>verbose</code>	whether to print progress messages

**Value**

`jaccard.test.mca` returns a list consisting of

<code>statistics</code>	centered Jaccard/Tanimoto similarity coefficient
<code>pvalue</code>	p-value
<code>expectation</code>	expectation

**Examples**

```
set.seed(1234)
x = rbinom(100,1,.5)
y = rbinom(100,1,.5)
jaccard.test.mca(x,y,accuracy = 1e-05)
```

---

```
jaccard.test.pairwise
```

*Pair-wise tests for Jaccard/Tanimoto similarity coefficients*

---

**Description**

Given a data matrix, it computes pair-wise Jaccard/Tanimoto similarity coefficients and p-values among rows (variables). Only for testing due to its use of a for-loop.

**Usage**

```
jaccard.test.pairwise(dat, method = "mca", verbose = TRUE,
  compute.qvalue = TRUE, ...)
```

**Arguments**

<code>dat</code>	a data matrix
<code>method</code>	a method to compute a p-value ("mca", "bootstrap", "asymptotic", or "exact")
<code>verbose</code>	whether to print progress messages
<code>compute.qvalue</code>	whether to compute q-values
<code>...</code>	optional arguments for specific computational methods

**Value**

`jaccard.test.pairwise` returns a list of matrices

`statistics`      Jaccard/Tanimoto similarity coefficients

`pvalues`        p-values

`qvalues`        q-values

**See Also**

[jaccard.test](#)

---

`runJaccardApp`

*Launch an interactive Shiny app on a local network*

---

**Description**

Launch an interactive Shiny app on a local network

**Usage**

`runJaccardApp()`

# Index

jaccard, 2  
jaccard.ev, 3  
jaccard.rahman, 3  
jaccard.test, 4, 10  
jaccard.test.asymptotic, 4, 5, 6  
jaccard.test.bootstrap, 4, 5, 7  
jaccard.test.exact, 4, 5, 8  
jaccard.test.mca, 4, 5, 8  
jaccard.test.pairwise, 9  
  
runJaccardApp, 10