

stringdist\_api

Generated by Doxygen 1.8.13



# Contents

- 1 Stringdist C API** **1**
  - 1.1 Using the stringdist C API . . . . . 1
  - 1.2 Character encoding . . . . . 1
  - 1.3 Thread safety . . . . . 1
  
- 2 File Index** **3**
  - 2.1 File List . . . . . 3
  
- 3 File Documentation** **5**
  - 3.1 stringdist\_api.h File Reference . . . . . 5
    - 3.1.1 Detailed Description . . . . . 5
    - 3.1.2 Function Documentation . . . . . 5
      - 3.1.2.1 sd\_amatch() . . . . . 6
      - 3.1.2.2 sd\_get\_qgrams() . . . . . 7
      - 3.1.2.3 sd\_lower\_tri() . . . . . 7
      - 3.1.2.4 sd\_soundex() . . . . . 8
      - 3.1.2.5 sd\_stringdist() . . . . . 9
  
- Index** **11**



# Chapter 1

## Stringdist C API

### Author

Mark van der Loo, Jan van der Laan, R Core Team, Paul Hsieh, Chris Muir

### Version

R package `stringdist` version 0.9.5.0 and higher.

### 1.1 Using the stringdist C API

To call the functions described here from your package you need to:

1. Make sure that `stringdist` is installed.
2. Add `stringdist` to `Imports` (or `Depends`) and `LinkingTo` in the `DESCRIPTION` file.
3. In your source file under the package's `/src` directory, add the line

```
#include <stringdist_api.h>
```

An example of a published package using this API is [refinr](#). A minimal example can be found [here](#).

### 1.2 Character encoding

All character vector input is expected to be in UTF-8 (this also allows ASCII). Distance computations are based on UTF [code points](#) unless `useBytes` is `TRUE`, in which case distances are computed over byte sequences. Using non-UTF-8 encoded strings is untested and is highly likely to result in errors.

### 1.3 Thread safety

It is not safe to call functions from `stringdist` C API from multiple concurrent threads.



# Chapter 2

## File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

[stringdist\\_api.h](#)  
Functions exported from the stringdist package . . . . . 5



## Chapter 3

# File Documentation

### 3.1 stringdist\_api.h File Reference

Functions exported from the stringdist package.

```
#include <R_ext/Rdynload.h>
```

#### Functions

- SEXP attribute\_hidden [sd\\_amatch](#) (SEXP x, SEXP table, SEXP method, SEXP nomatch, SEXP matchNA, SEXP weight, SEXP p, SEXP bt, SEXP q, SEXP maxDistance, SEXP useBytes, SEXP nthrd)  
*Find the location of values in x in table by approximate matching.*
- SEXP attribute\_hidden [sd\\_get\\_qgrams](#) (SEXP a, SEXP qq)  
*Compute q-gram counts.*
- SEXP attribute\_hidden [sd\\_lower\\_tri](#) (SEXP a, SEXP method, SEXP weight, SEXP p, SEXP bt, SEXP q, SEXP useBytes, SEXP nthrd)  
*Lower tridiagonal elements of distance matrix.*
- SEXP attribute\_hidden [sd\\_soundex](#) (SEXP x, SEXP useBytes)  
*Compute soundex code.*
- SEXP attribute\_hidden [sd\\_stringdist](#) (SEXP a, SEXP b, SEXP method, SEXP weight, SEXP p, SEXP bt, SEXP q, SEXP useBytes, SEXP nthrd)  
*compute string distances*

#### 3.1.1 Detailed Description

Functions exported from the stringdist package.

#### 3.1.2 Function Documentation

3.1.2.1 `sd_amatch()`

```
SEXP attribute_hidden sd_amatch (
    SEXP x,
    SEXP table,
    SEXP method,
    SEXP nomatch,
    SEXP matchNA,
    SEXP weight,
    SEXP p,
    SEXP bt,
    SEXP q,
    SEXP maxDistance,
    SEXP useBytes,
    SEXP nthrd )
```

Find the location of values in `x` in `table` by approximate matching.

## Parameters

<code>x</code>	[character] vector.
<code>table</code>	[character] vector (lookup table)
<code>method</code>	[integer] scalar, indicating the distance method as follows <ul style="list-style-type: none"> <li>• 0: Optimal String Alignment ("osa")</li> <li>• 1: Levenshtein ("lv")</li> <li>• 2: Damerau-Levenshtein ("dl")</li> <li>• 3: Hamming ("hamming")</li> <li>• 4: Longest Common Substring ("lcs")</li> <li>• 5: q-gram ("qgram")</li> <li>• 6: cosine ("cosine")</li> <li>• 7: Jaccard ("jaccard")</li> <li>• 8: Jaro-Winkler ("jw")</li> <li>• 9: Soundex ("soundex")</li> </ul>
<code>nomatch</code>	[integer] The value to be returned when no match is found.
<code>matchNA</code>	Should NAs be matched? Default behaviour mimics the behaviour of base <code>match</code> , meaning that NA matches NA (see also the note on NA handling below).
<code>weight</code>	[numeric] vector. Edit penalty For <code>method='osa'</code> or <code>'dl'</code> , the penalty for deletion, insertion, substitution and transposition, in that order. When <code>method='lv'</code> , the penalty for transposition is ignored. When <code>method='jw'</code> , the weights associated with characters of <code>a</code> , characters from <code>b</code> and the transposition weight, in that order. Weights must be positive and not exceed 1. <code>weight</code> is ignored completely for other methods
<code>q</code>	[integer] scalar. Size of the q-gram; must be nonnegative. Only applies to <code>method='qgram', 'jaccard'</code> or <code>'cosine'</code> .
<code>maxDistance</code>	[numeric] scalar. The maximum distance allowed for matching.
<code>p</code>	[numeric] scalar. Penalty factor for Jaro-Winkler distance. The valid range for <code>p</code> is $0 \leq p \leq 0.25$ . If <code>p=0</code> (default), the Jaro-distance is returned. Applies only to <code>method='jw'</code> .
<code>bt</code>	[numeric] vector. Winkler's boost threshold. Winkler's penalty factor is only applied when the Jaro distance is larger than <code>bt</code> . Applies only to <code>method='jw'</code> and <code>p&gt;0</code> .

## Parameters

<i>useBytes</i>	Perform byte-wise comparison (i.e. do not translate UTF-8 to integer prior to distance calculation)
<i>nthread</i>	[integer] scalar. Maximum number of threads to use.

## Returns

[integer] vector of length(x) with indices in table.

## 3.1.2.2 sd\_get\_qgrams()

```
SEXP attribute_hidden sd_get_qgrams (
    SEXP a,
    SEXP qq )
```

Compute q-gram counts.

## Parameters

<i>a</i>	[character] vector
<i>qq</i>	[integer] scalar.

## Returns

A [numeric] vector of length(a)\*n\_qgrams, where n\_qgrams is the number of different qgrams observed in the elements of a. The output vector has an attribute called qgrams, which is an integer vector of size q\*n\_qgrams containing integer (UTF-32) labels for the q-grams sequentially.

## 3.1.2.3 sd\_lower\_tri()

```
SEXP attribute_hidden sd_lower_tri (
    SEXP a,
    SEXP method,
    SEXP weight,
    SEXP p,
    SEXP bt,
    SEXP q,
    SEXP useBytes,
    SEXP nthrd )
```

Lower tridiagonal elements of distance matrix.

## Parameters

<i>a</i>	[character] vector
----------	--------------------

## Parameters

<i>method</i>	[integer] scalar, indicating the distance method as follows <ul style="list-style-type: none"> <li>• 0: Optimal String Alignment ("osa")</li> <li>• 1: Levenshtein ("lv")</li> <li>• 2: Damerau-Levenshtein ("dl")</li> <li>• 3: Hamming ("hamming")</li> <li>• 4: Longest Common Substring ("lcs")</li> <li>• 5: q-gram ("qgram")</li> <li>• 6: cosine ("cosine")</li> <li>• 7: Jaccard ("jaccard")</li> <li>• 8: Jaro-Winkler ("jw")</li> <li>• 9: Soundex ("soundex")</li> </ul>
<i>weight</i>	[numeric] vector. Edit penalty For <code>method='osa'</code> or <code>'dl'</code> , the penalty for deletion, insertion, substitution and transposition, in that order. When <code>method='lv'</code> , the penalty for transposition is ignored. When <code>method='jw'</code> , the weights associated with characters of <code>a</code> , characters from <code>b</code> and the transposition weight, in that order. Weights must be positive and not exceed 1. <code>weight</code> is ignored completely for other methods
<i>q</i>	[integer] scalar. Size of the q-gram; must be nonnegative. Only applies to <code>method='qgram', 'jaccard' or 'cosine'</code> .
<i>p</i>	[numeric] scalar. Penalty factor for Jaro-Winkler distance. The valid range for <code>p</code> is $0 \leq p \leq 0.25$ . If <code>p=0</code> (default), the Jaro-distance is returned. Applies only to <code>method='jw'</code> .
<i>bt</i>	[numeric] vector. Winkler's boost threshold. Winkler's penalty factor is only applied when the Jaro distance is larger than <code>bt</code> . Applies only to <code>method='jw'</code> and <code>p&gt;0</code> .
<i>useBytes</i>	Perform byte-wise comparison (i.e. do not translate UTF-8 to integer prior to distance calculation)
<i>nthread</i>	[integer] scalar. Maximum number of threads to use.

## Returns

A [numeric] vector of length  $n*(n-1)/2$ , where  $n=length(a)$ . It contains the positive values of consecutive columns of the distance matrix. Also see the R-code in `stringdist:::lower_tri`.

## 3.1.2.4 sd\_soundex()

```
SEXP attribute_hidden sd_soundex (
    SEXP x,
    SEXP useBytes )
```

Compute soundex code.

## Parameters

in	<code>x</code>	[character] vector
in	<code>useBytes</code>	[logical] scalar.

## Returns

A list with length(x) element. Each element is a length 4 integer vector representing a 4-character soundex code. The integers are ASCII code points.

## 3.1.2.5 sd\_stringdist()

```
SEXP attribute_hidden sd_stringdist (
    SEXP a,
    SEXP b,
    SEXP method,
    SEXP weight,
    SEXP p,
    SEXP bt,
    SEXP q,
    SEXP useBytes,
    SEXP nthrd )
```

compute string distances

## Parameters

<i>a</i>	[character] vector
<i>b</i>	[character] vector
<i>method</i>	[integer] scalar, indicating the distance method as follows <ul style="list-style-type: none"> <li>• 0: Optimal String Alignment ("osa")</li> <li>• 1: Levenshtein ("lv")</li> <li>• 2: Damerau-Levenshtein ("dl")</li> <li>• 3: Hamming ("hamming")</li> <li>• 4: Longest Common Substring ("lcs")</li> <li>• 5: q-gram ("qgram")</li> <li>• 6: cosine ("cosine")</li> <li>• 7: Jaccard ("jaccard")</li> <li>• 8: Jaro-Winkler ("jw")</li> <li>• 9: Soundex ("soundex")</li> </ul>
<i>weight</i>	[numeric] vector. Edit penalty For method='osa' or 'dl', the penalty for deletion, insertion, substitution and transposition, in that order. When method='lv', the penalty for transposition is ignored. When method='jw', the weights associated with characters of a, characters from b and the transposition weight, in that order. Weights must be positive and not exceed 1. weight is ignored completely for other methods
<i>q</i>	[integer] scalar. Size of the q-gram; must be nonnegative. Only applies to method='qgram', 'jaccard' or 'cosine'.
<i>p</i>	[numeric] scalar. Penalty factor for Jaro-Winkler distance. The valid range for p is 0 <= p <= 0.25. If p=0 (default), the Jaro-distance is returned. Applies only to method='jw'.
<i>bt</i>	[numeric] vector. Winkler's boost threshold. Winkler's penalty factor is only applied when the Jaro distance is larger than bt. Applies only to method='jw' and p>0.
<i>useBytes</i>	Perform byte-wise comparison (i.e. do not translate UTF-8 to integer prior to distance calculation)
<i>nthread</i>	[integer] scalar. Maximum number of threads to use.

**Returns**

A `[numeric]` vector of length `max(length(a), length(b))` where the shortest vector is recycled over the longer (no warnings are given when the longer length is not an integer multiple of the shorter length).

# Index

- sd\_amatch
  - stringdist\_api.h, [5](#)
- sd\_get\_qgrams
  - stringdist\_api.h, [7](#)
- sd\_lower\_tri
  - stringdist\_api.h, [7](#)
- sd\_soundex
  - stringdist\_api.h, [8](#)
- sd\_stringdist
  - stringdist\_api.h, [9](#)
- stringdist\_api.h, [5](#)
  - sd\_amatch, [5](#)
  - sd\_get\_qgrams, [7](#)
  - sd\_lower\_tri, [7](#)
  - sd\_soundex, [8](#)
  - sd\_stringdist, [9](#)