Package ‘hashids’

August 29, 2016

Title Generate Short Unique YouTube-Like IDs (Hashes) from Integers

Version 0.9.0

Description An R port of the hashids library. hashids generates YouTube-like hashes from integers or vector of integers. Hashes generated from integers are relatively short, unique and non-sequential. hashids can be used to generate unique ids for URLs and hide database row numbers from the user. By default hashids will avoid generating common English curse-words by preventing certain letters being next to each other. hashids are not one-way: it is easy to encode an integer to a hashid and decode a hashid back into an integer.


BugReports https://github.com/ALShum/hashids-r/issues

Depends R (>= 3.2.2)

License MIT + file LICENSE

LazyData true

Suggests testthat

NeedsCompilation no

Author Alex Shum [aut, cre],
Ivan Akimov [aut] (original author of hashids -- implemented in javascript),
David Aurelio [ctb] (implemented hashids in python 2 and 3)

Maintainer Alex Shum <Alex@ALShum.com>

Repository CRAN

Date/Publication 2015-09-11 10:10:26

R topics documented:

ascii_val .................................................. 2
base16_to_dec ........................................... 2
decode .................................................... 3
decode_hex ................................................. 3
dec_to_base16 ............................................. 4
encode ..................................................... 4
ascii_val

**Description**

Calculate the ascii value number of a character

**Usage**

```r
ascii_val(char)
```

**Arguments**

- `char`: character

**Value**

ascii value integer

base16_to_dec

**Description**

Converts a base 16 string to a base 10 number. Because I couldn’t get base R functions to work for big hex numbers.

**Usage**

```r
base16_to_dec(str_16)
```

**Arguments**

- `str_16`: base 16 number as a string.
**decode**

*Description*

Decodes a hashid into the original integer or integer vector

*Usage*

```
decode(hash_str, settings)
```

*Arguments*

- `hash_str`: hashid string to decode into integer or integer vector
- `settings`: Settings list generated by hashid_settings

*Value*

integer or integer vector

---

**decode_hex**

*Description*

Decodes a hashid into the original hexadecimal number

*Usage*

```
decode_hex(hashid, settings)
```

*Arguments*

- `hashid`: hashid to decode
- `settings`: Settings list generated by hashid_settings

*Value*

hexadecimal number as a string
### dec_to_base16

*Converts a base 10 number to base 16 number. Because I couldn’t get R’s as.hexmode() to work for big integers.*

**Description**

Converts a base 10 number to base 16 number. Because I couldn’t get R’s as.hexmode() to work for big integers.

**Usage**

```
dec_to_base16(dec)
```

**Arguments**

- `dec` base 10 integer

**Value**

base 16 number as a string

---

### encode

*Encodes an integer or integer vector into a hashid string. All numbers must be non-negative integers.*

**Description**

Encodes an integer or integer vector into a hashid string. All numbers must be non-negative integers.

**Usage**

```
encode(int, settings)
```

**Arguments**

- `int` Integer or integer vector to encode
  - `settings` Settings list generated by hashid_settings

**Value**

hashid string
**encode_hex**

*Encodes a hexadecimal number into a hashid*

**Description**

Encodes a hexadecimal number into a hashid

**Usage**

```python
encode_hex(hex_str, settings)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hex_str</code></td>
<td>Hexadecimal number as string</td>
</tr>
<tr>
<td><code>settings</code></td>
<td>Settings list generated by hashid_settings</td>
</tr>
</tbody>
</table>

**Value**

hashid string

**enforce_min_length**

*Enforces hashid minimum length by padding the hashid with additional characters.*

**Description**

Enforces hashid minimum length by padding the hashid with additional characters.

**Usage**

```python
enforce_min_length(encoded, min_length, alphabet, guards, values_hash)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>encoded</code></td>
<td>encoded hashid</td>
</tr>
<tr>
<td><code>min_length</code></td>
<td>minimum length required for hashid</td>
</tr>
<tr>
<td><code>alphabet</code></td>
<td>set of letters used to generate hashid</td>
</tr>
<tr>
<td><code>guards</code></td>
<td>set of guards used to generate hashid</td>
</tr>
<tr>
<td><code>values_hash</code></td>
<td>value hashed used to select guard characters</td>
</tr>
</tbody>
</table>

**Value**

hashid with padded characters to insure minimum length
**hash**

Maps an integer to a string. Generated string will be inversely proportional to alphabet length.

**Description**

Maps an integer to a string. Generated string will be inversely proportional to alphabet length.

**Usage**

```python
hash(number, alphabet)
```

**Arguments**

- `number` Integer to hash
- `alphabet` Possible letters for string.

**Value**

hashed string

---

**hashid_defaults**

Default Values for hashid settings

**Description**

Default alphabet, separators, and ratio of character separators and guards for hashid

**Usage**

```
DEFAULT_ALPHABET
DEFAULT_SEPS
RATIO_SEPARATORS
RATIO_GUARDS
```

**Format**

```
chr "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890"
```

**Source**

http://www.hashids.org
hashid_settings  A function to create a hashid settings list.

Description
A function to create a hashid settings list.

Usage
hashid_settings(salt, min_length = 0, alphabet = DEFAULT_ALPHABET, 
sep = DEFAULT_SEPS)

Arguments
- salt: An additional string to make hashids more unique.
- min_length: Minimum length for hashid.
- alphabet: String of characters for hashid.
- sep: String of characters to use as separators.

Value
A list of parameters used in encoding and decoding.

shuffle  Permutations the characters in a string based on an inputted salt string.

Description
Permutations the characters in a string based on an inputted salt string.

Usage
shuffle(string, salt)

Arguments
- string: String to be permuted
- salt: cryptograph salt string that is used to permute strings

Value
shuffled string
split

*Splits a string based on a set of splitting characters*

Description

Splits a string based on a set of splitting characters

Usage

`split(string, splitters)`

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>String to split</td>
</tr>
<tr>
<td>splitters</td>
<td>set of splitting characters as a string</td>
</tr>
</tbody>
</table>

Value

split vector of characters

unhash

*Unhashes a string to an integer based on alphabet.*

Description

Unhashes a string to an integer based on alphabet.

Usage

`unhash(hashed, alphabet)`

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashed</td>
<td>String to unhash</td>
</tr>
<tr>
<td>alphabet</td>
<td>Set of letters used for hashing</td>
</tr>
</tbody>
</table>

Value

Unhashed integer
Index

*Topic **datasets**
  hashid_defaults, 6

ascii_val, 2

base16_to_dec, 2

dec_to_base16, 4
decode, 3
decode_hex, 3
DEFAULT_ALPHABET (hashid_defaults), 6
DEFAULT SEPS (hashid_defaults), 6

encode, 4
encode_hex, 5
enforce_min_length, 5

hash, 6
hashid_defaults, 6
hashid_settings, 7

RATIO_GUARDS (hashid_defaults), 6
RATIO SEPARATORS (hashid_defaults), 6

shuffle, 7
split, 8

unhash, 8