Package ‘squids’

May 30, 2025

Title Short Quasi-Unique Identifiers (SQUIDs)

Version 25.5.6

Description It is often useful to produce short, quasi-unique identifiers (SQUIDs) without the benefit of a central authority to prevent duplication. Although Universally Unique Identifiers (UUIDs) provide for this, these are also unwieldy; for example, the most used UUID, version 4, is 36 characters long. SQUIDs are short (8 characters) at the expense of having more collisions, which can be mitigated by combining them with human-produced suffixes, yielding relatively brief, half human-readable, almost-unique identifiers (see for example the identifiers used for Decentralized Construct Taxonomies; Peters & Crutzen, 2024 <doi:10.15626/MP.2022.3638>). SQUIDs are the number of centiseconds elapsed since the beginning of 1970 converted to a base 30 system. This package contains functions to produce SQUIDs as well as convert them back into dates and times.

License GPL (>= 3)

BugReports `https://codeberg.org/R-packages/squids/issues`

URL `https://squids.opens.science`

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Gjalt-Jorn Peters [aut, cre] (ORCID: <`https://orcid.org/0000-0002-0336-9589`>)

Maintainer Gjalt-Jorn Peters <`squids@opens.science`>

Repository CRAN

Date/Publication 2025-05-30 09:40:02 UTC

Contents

`base30toNumeric` *Conversion between base10 and base30*

Description

The conversion functions from base10 to base30 and vice versa are used by the `squids()` functions.

Usage

```
base30toNumeric(x)

numericToBase30(x)
```

Arguments

`x`	The vector to convert (numeric for `numericToBase30`, character for `base30toNumeric`).

Details

The symbols to represent the 'base 30' system are the 0-9 followed by the alphabet without vowels but including the y. This vector is available as `base30`.

Value

The converted vector (numeric for `base30toNumeric`, character for `numericToBase30`).

Examples

```
squids::numericToBase30(
  654321
);

squids::base30toNumeric(
  squids::numericToBase30(
    654321
  )
);
```

`cat0` *Concatenate to screen without spaces*

Description

The cat0 function is to cat what paste0 is to paste; it simply makes concatenating many strings without a separator easier.

Usage

```
cat0(..., sep = "")
```

Arguments

`...`	The character vector(s) to print; passed to `cat()`.
`sep`	The separator to pass to `cat()`, of course, `""` by default.

Value

Nothing (invisible `NULL`, like `cat()`).

Examples

```
cat0("The first variable is '", names(mtcars)[1], "'.");
```

`highest_squid` *Finding extreme (highest or lowest) SQUIDs*

Description

Finding extreme (highest or lowest) SQUIDs

Usage

```
highest_squid(x)

lowest_squid(x)
```

Arguments

`x`	A vector of SQUIDs (or a list of vectors, which will be recursively `unlist()`ed).

Value

The highest or lowest SQUID

Examples

```
squids::highest_squid(
  squids::squids(5)
);

squids::lowest_squid(
  squids::squids(5)
);
```

---

squids *Generate short quasi-unique identifiers (SQUIDs)*

Description

This function generates short quasi-unique identifiers.

Usage

```
squids(x, origin = Sys.time(), follow = NULL, followBy = NULL)

timestamp_to_squids(x)
```

Arguments

`x`	The number of identifiers to generate.
`origin`	The origin to use when generating the actual identifiers. These identifiers are the present UNIX timestamp (i.e. the number of seconds elapsed since the UNIX epoch, the first of january 1970), accurate to two decimal places (i.e. to centiseconds), converted to the base 30 system using `numericToBase30()`. By default, the present time is used as origin, one one centisecond is added for every identifiers to generate. `origin` can be set to other values to work with different origins (of course, don't use this unless you understand very well what you're doing!).
`follow`	A vector of one or more SQUIDs (or a list; lists are recursively `unlist()`ed); the highest SQUID will be taken, converted to a timestamp, and used as `origin` (well, 0.01 second later), so that the new SQUIDs will follow that sequence.
`followBy`	When following a vector of SQUIDs, this can be used to specify the distance between the two vectors in centiseconds.

Details

SQUIDs are defined as 8-character strings that express a timestamp (the number of centiseconds that passed since the UNIX Epoch) in a base 30 decimal system. The lowest possible SQUID, therefore, is `00000001` (which corresponds to 1970-01-01 00:00:00 UTC), and the highest possible SQUID is `zzzzzzzz`, which corresponds to 2177-11-28 11:59:59 UTC.

Value

A vector of SQUIDs.

Examples

```
exampleSQUIDs <-
  squids::squids(5);

### Show how SQUIDs are the converted date/time
squids::squids_to_datetime(
  exampleSQUIDs
);

### These seem the same, but if we take these as
### timestamps (seconds passed since the UNIX Epoch)
### and multiply with 100 to see the centiseconds,
### we see the differences:
as.numeric(
  squids::squids_to_datetime(
    exampleSQUIDs
  )
) * 100;

### Get a sequence following the first one
squids::squids(5, follow=exampleSQUIDs);

### Follow at a distance
squids::squids(
  5,
  follow=exampleSQUIDs,
  followBy = 3
);
```

`squids_to_datetime` *Converting SQUIDs back to timestamps and dates/times*

Description

Converting SQUIDs back to timestamps and dates/times

Usage

```
squids_to_datetime(x, tz = "UTC")

squids_to_timestamp(x)
```

Arguments

`x`	A vector of one or more SQUIDs
`tz`	The timezone to use

Value

A vector of one or more timestamps or `POSIXct` date/time objects

Examples

```
exampleSQUID <-
  squids::squids();

### Timestamp (second since UNIX Epoch,
###            1970-01-01, 00:00:00 UTC)
squids::squids_to_timestamp(
  exampleSQUID
);

squids::squids_to_datetime(
  exampleSQUID
);

### In Central European Time
squids::squids_to_datetime(
  exampleSQUID,
  tz = "CET"
);
```

`vecTxt` *Easily parse a vector into a character value*

Description

Easily parse a vector into a character value

Usage

```
vecTxt(
  vector,
  delimiter = ", ",
  useQuote = "",
  firstDelimiter = NULL,
  lastDelimiter = " & ",
  firstElements = 0,
  lastElements = 1,
  lastHasPrecedence = TRUE
)

vecTxtQ(vector, useQuote = "'", ...)
```

Arguments

`vector`	The vector to process.
`delimiter`, `firstDelimiter`, `lastDelimiter`	The delimiters to use for respectively the middle, first `firstElements`, and last `lastElements` elements.
`useQuote`	This character string is pre- and appended to all elements; so use this to quote all elements (`useQuote="'"`), doublequote all elements (`useQuote='"'`), or anything else (e.g. `useQuote='|'`). The only difference between `vecTxt` and `vecTxtQ` is that the latter by default quotes the elements.
`firstElements`, `lastElements`	The number of elements for which to use the first respective last delimiters
`lastHasPrecedence`	If the vector is very short, it's possible that the sum of firstElements and lastElements is larger than the vector length. In that case, downwardly adjust the number of elements to separate with the first delimiter (`TRUE`) or the number of elements to separate with the last delimiter (`FALSE`)?
`...`	Any addition arguments to `vecTxtQ` are passed on to `vecTxt`.

Value

A character vector of length 1.

Examples

```
vecTxtQ(names(mtcars));
```

Index