Package 'squids'

June 7, 2025

Title Short Quasi-Unique Identifiers (SQUIDs)

Version 25.6.1

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 2025-06-07 10:50:02 UTC

Contents

`add_squids_to_df` *Add a column with SQUIDs to a data frame*

Description

Add a column with SQUIDs to a data frame

Usage

```
add_squids_to_df(
  x,
  colName = "SQUID",
  warnAgainstOverwriting = TRUE,
  origin = Sys.time(),
  follow = NULL,
  followBy = NULL
)
```

Arguments

`x`	The data frame
`colName`	The name of the column to add; set to `NULL` to return the column instead of the data frame.
`warnAgainstOverwriting`	Whether to throw an error if a column with name `colName` already exists.

origin	The origin to use when generating the SQUIDs. This allows you to reproduce the same sequence of SQUIDs. You can easily get an origin with get_current_origin(). The origin is a timestamp; an object of class POSIXct (see squids-package for more details).
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.

Value

If colName = NULL, the column with SQUIDs; otherwise, x with an additional column named with the value of colName.

Examples

```
squids::add_squids_to_df(
  mtcars
);
```

---

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], "'.");
```

checkPkgs *Check for presence of a package*

Description

This function efficiently checks for the presence of a package without loading it (unlike `library()` or `require()`. This is useful to force yourself to use the `package::function()` syntax for addressing functions; you can make sure required packages are installed, but their namespace won't attach to the search path.

Usage

```
checkPkgs(
  ...,
  install = FALSE,
  load = FALSE,
  repos = "https://cran.rstudio.com"
)
```

Arguments

`...`	A series of packages. If the packages are named, the names are the package names, and the values are the minimum required package versions (see the second example).
`install`	Whether to install missing packages from `repos`.
`load`	Whether to load packages (which is exactly *not* the point of this function, but hey, YMMV).
`repos`	The repository to use if installing packages; default is the RStudio repository.

Value

Invisibly, a vector of the available packages.

Examples

```
squids::checkPkgs('base');

### Require a specific version
squids::checkPkgs(squids = "25.1.1");

### This will show the error message
tryCatch(
  squids::checkPkgs(
    base = "99",
    stats = "42.5",
    squids = 100
  ),
```

```
  error = print
);
```

`extract_squids` *Extract SQUIDs from a character vector*

Description

This function simply looks for matches with

Usage

```
extract_squids(x)
```

Arguments

x	The character vector

Value

A character vector with SQUIDs

Examples

```
example <-
  paste0(
    "some prefix text ", 1:5, " ",
    squids::squids(5),
    " ", letters[1:5], " some suffix text"
  );

squids::extract_squids(
  example
);
```

`get_current_origin` *Get the current origin for each reuse*

Description

Get the current origin for each reuse

Usage

```
get_current_origin(
  as = "time",
  suppressPrinting = FALSE,
  format = "%Y-%m-%d %H:%M:%S %Z"
)
```

Arguments

as	Whether to return the origin as `character` value (can also be specified by passing `string` or `text`), as a `numeric` value (can also be specified by passing `number`), or as `POSIX` time (can also be specified by passing `time`).
suppressPrinting	Whether to suppress printing the message about how to store the origin in your R script.
format	If returning `character`, the `format` to pass to `base::format()` when formatting the time to a character value

Value

The origin, in the format specified in `as`.

Examples

```
squids::get_current_origin();

squids::get_current_origin(
  as = "number"
);

squids::get_current_origin(
  as = "time"
);
```

---

get_squid *Set or get a SQUID (to follow)*

Description

Because the SQUID is saved in the options, it persists when changed e.g. in function calls, for example when using `lapply()`.

Usage

```
get_squid(namespace = NULL)

set_squid(x, namespace = NULL)
```

Arguments

namespace	Optionally, for saving multiple SQUIDs, a namespace.
x	A SQUID (or several; the highest is stored).

Value

The saved SQUID.

Invisibly, x.

Examples

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

squids::set_squid(
  exampleSQUID
);

squids::get_squid();
```

---

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)
);
```

`next_squid` *Get the next SQUID (or SQUIDs)*

Description

Get the next SQUID (or SQUIDs)

Usage

```
next_squid(x, n = 1, followBy = NULL)
```

Arguments

`x`	The SQUID or SQUIDs to follow (`follow` in the `squids()` function).
`n`	The number of following SQUIDs you want
`followBy`	When following a vector of SQUIDs, this can be used to specify the distance between the two vectors in centiseconds.

Value

One or more SQUIDs

Examples

```
exampleSQUID <-
  squids::squids(1);

exampleSQUID;

squids::next_squid(exampleSQUID);

### Or for multiple SQUIDs
exampleSQUIDs <-
  squids::squids(5);

exampleSQUIDs;

squids::next_squids(exampleSQUIDs, 3);
```

`opts` *Options for the squids package*

Description

The `squids::opts` object contains three functions to set, get, and reset options used by the zirconia package. Use `squids::opts$set` to set options, `squids::opts$get` to get options, or `squids::opts$reset` to reset specific or all options to their default values.

Usage

```
opts
```

Format

An object of class `list` of length 4.

Details

It is normally not necessary to get or set `squids` options.

The following arguments can be passed:

... For `squids::opts$set`, the dots can be used to specify the options to set, in the format `option = value`, for example, `utteranceMarker = "\n"`. For `squids::opts$reset`, a list of options to be reset can be passed.

option For `squids::opts$set`, the name of the option to set.

default For `squids::opts$get`, the default value to return if the option has not been manually specified.

The following options can be set:

silent Whether to be silent or chatty.

encoding The default encoding when reading or writing files.

preventOverwriting Whether to be prevent overwriting of existing files.

debug Sometimes used to display debugging information.

Examples

```
### Get the default 'silent' setting
squids::opts$get(silent);

### Set it to FALSE (to be chatty)
squids::opts$set(silent = FALSE);

### Check that it worked
squids::opts$get(silent);
```

```
### Reset this option to its default value
squids::opts$reset(silent);

### Check that the reset worked, too
squids::opts$get(silent);
```

`orcid_to_shorcid` *Converting ORCIDs to ShORCIDs and vice versa*

Description

These functions produce ShORCIDs (Short ORCIDs) from ORCIDs and vice versa.

Usage

```
orcid_to_shorcid(x)

shorcid_to_orcid(x, url = FALSE)
```

Arguments

`x`	The ORCID(s) or ShORCID(s).
`url`	Whether to also return the ORCID or the ORCID URL (including the preceding "https://orcid.org/" bit)

Details

Conversion ORCID to ShORCID occurs by detaching the last character (the checksum) and storing it. Then in the first string of characters, all non-numbers are removed and the resulting number is converted to a base 30 system with `numericToBase30()`. The checksum is then re-attached. This is done separately because the checksum can be X (i.e. the only character in an ORCID that's not necessarily numeric). Then, an 'i' is prepended to ensure that the ShORCID starts with a letter. Conversion the other way around just inverts the process (and so uses `base30toNumeric()`).

Value

The ShORCID(s) or ORCID(s), as a character vector.

Examples

```
squids::orcid_to_shorcid(
  "0000-0002-9540-5371"
);
squids::shorcid_to_orcid(
  "i16g2sk1"
);
```

origin_to_squids *Convert an origin (a timestamp) to a SQUID*

Description

For convenience, `origin_to_squids()`, `datetime_to_squids()`, and `POSIXt_to_squids()` can all be used (i.e. they're aliases for the same function). For more information, see `squids()`. For conversion the other way around, see `squids_to_origin()`.

Usage

```
origin_to_squids(x)
```

Arguments

`x`	The time, e.g. `Sys.time()`.

Value

A SQUID

Examples

```
squids::origin_to_squids(
  Sys.time()
);
```

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

Description

The `squids::squids()` function generates a sequence of short quasi-unique identifiers (see squids-package for more details). The `squids::random_squids()` function is a convenience function that randomizes the result before returning it.

Usage

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

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

Arguments

x	The number of identifiers to generate.
origin	The origin to use when generating the SQUIDs. This allows you to reproduce the same sequence of SQUIDs. You can easily get an origin with `get_current_origin()`. The `origin` is a timestamp; an object of class `POSIXct` (see squids-package for more details).
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 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. More details are in the squids-package manual page.

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
);
```

`repeatStr` *Repeat a string a number of times*

Description

Repeat a string a number of times

Usage

```
repeatStr(n = 1, str = " ")
```

Arguments

`n`, `str`	Normally, respectively the frequency with which to repeat the string and the string to repeat; but the order of the inputs can be switched as well.

Value

A character vector of length 1.

Examples

```
### 10 spaces:
repStr(10);

### Three euro symbols:
repStr("\u20ac", 3);
```

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

Description

`squids_to_timestamp()` converts a SQUID back to a timestamp (the number of seconds that passed since the UNIC Epoch, 1970-01-01, 00:00:00 UTC), and `squids_to_datetime()`, `squids_to_origin()`, `squids_to_POSIXt()` convert a SQUID to a POSIX time object (which is why they also have a `tz` argument).

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"
);
```

`timestamp_to_squids` *Convert a timestamp to a SQUID*

Description

Convert a timestamp (the number of seconds that passed since the first of January, 1970) to a SQUID.

Usage

```
timestamp_to_squids(x)
```

Arguments

x	The timestamp (or a vector of timestamps)

Value

The SQUID(s)

Examples

```
timestamp <-
  as.numeric(Sys.time());

squids::timestamp_to_squids(
  timestamp
);
```

`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