Package: xlamisc (via r-universe)

June 1, 2026

Title Helper Functions for the XLA Ecosystem

Version 0.3.0

Description Helper functions for the XLA ecosystem.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports cli, checkmate, R6

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://r-xla.r-universe.dev

Date/Publication 2026-06-01 13:05:12 UTC

RemoteUrl https://github.com/r-xla/xlamisc

RemoteRef v0.3.0

RemoteSha fc062dca66e81cde09f580d17960b3f5f7e0bf52

Contents

xlamisc-package *xlamisc: Helper Functions for the XLA Ecosystem*

Description

Helper functions for the XLA ecosystem.

Author(s)

Maintainer: Sebastian Fischer <seb.fischer@tutamail.com> (ORCID)

Authors:

- Daniel Falbel <daniel@posit.co> (ORCID)

format_bib *Format Bibentries in Roxygen*

Description

Operates on a named list of `bibentry()` entries and formats them for documentation with **roxygen2**.

- `format_bib()` is intended to be called in the `@references` section and formats the complete entry using `tools::toRd()`.
- `cite_bib()` returns a short inline citation in the format `"LastName (Year)"`.

Usage

```
format_bib(..., bibentries = NULL)

cite_bib(..., bibentries = NULL)
```

Arguments

`...`	(`character()`) One or more names of bibentries.
`bibentries`	(named `list()`) Named list of bibentries. If `NULL`, looks up an object called `bibentries` in the calling environment.

Value

(`character(1)`).

Examples

```
bibentries = list(R = citation())
format_bib("R")
cite_bib("R")
```

---

get_dims *Get the dimensions of a data object*

Description

Get the "dimension" of an R array or vector.

Usage

```
get_dims(data)
```

Arguments

data	(any) The data object to get the dimensions of.

Value

(integer())

LRUCache *Least Recently Used Cache*

Description

An implementation of a least-recently-used cache built on top of utils::hashtab. Therefore, arbitrary keys can be used, as opposed to the implementation in **cachemem**, which relies on environments.

Details

This LRU cache is implemented as a combination of a hashmap and a doubly linked list. The hashmap is used for lookups and the doubly linked list is used to maintain the ordering of most recently used to least recently used items. Whenever an element is added or accessed, it is moved to the front of the list.

Active bindings

capacity (integer(1))
: The maximum capacity of the cache.

size (integer(1))
: The number of items currently stored.

Methods

Public methods:

- LRUCache$new()
- LRUCache$get()
- LRUCache$set()
- LRUCache$has()
- LRUCache$remove()
- LRUCache$clear()
- LRUCache$keys_mru_to_lru()
- LRUCache$clone()

Method `new()`**:** Initialize a new LRU cache with a given `capacity`.

Usage:

```
LRUCache$new(capacity)
```

Arguments:

`capacity (integer(1))`
Number of items the cache holds.

Method `get()`**:** Get the value for `key` and mark it as most-recently-used.

Usage:

```
LRUCache$get(key, default = NULL)
```

Arguments:

`key (any)`
Key.

`default (any)`
Default value returned by `get()` when `key` is not present.

Method `set()`**:** Set the value for `key`, updating recency and evicting LRU as needed.

Usage:

```
LRUCache$set(key, value)
```

Arguments:

`key (any)`
Key.

`value (any)`
Value

Method `has()`**:** Check whether a given `key` exists in the cache.

Usage:

```
LRUCache$has(key)
```

Arguments:

`key (any)`
Key.

Method `remove()`**:** Remove `key` from the cache, returning the value if it was present or `NULL` otherwise.

Usage:

```
LRUCache$remove(key)
```

Arguments:

`key` (any)
Key.

Method `clear()`**:** Clear all entries from the cache.

Usage:

```
LRUCache$clear()
```

Method `keys_mru_to_lru()`**:** Return keys ordered from most-recently-used to least-recently-used.

Usage:

```
LRUCache$keys_mru_to_lru()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LRUCache$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`new_list_of` *Create a new list of class*

Description

Returns a constructor for a typed list. The constructor checks that `items` is a list and runs the optional `validator`, but does not iterate `items` to validate per-element class – this makes it cheap enough to call from hot paths like IR construction. Trust the caller to pass items of the right class; downstream consumers will fail with their own (clearer) error if not.

Usage

```
new_list_of(class_name, item_class, validator = NULL)
```

Arguments

`class_name`	The name of the class
`item_class`	The class of the items in the list. Documentation only – not enforced at construction.
`validator`	A validator function. Receives `items`, returns `NULL` on success or a `cli_abort`-style message on failure.

Value

A list-of-`class_name` constructor.

seq0 *Sequence of length 0*

Description

Like `seq_len` and `seq_along` but starts at 0.

Usage

```
seq_len0(n)

seq_along0(x)
```

Arguments

`n`	(`integer(1)`) The length of the sequence.
`x`	(`integer()`) The vector to sequence along.

Value

(`integer()`)

shapevec_repr *Format shape vector as string*

Description

Formats a shape vector as a string like `(2,3,4)`. NA values are replaced with `?`.

Usage

```
shapevec_repr(shape)
```

Arguments

`shape`	(`integer()`) A shape vector.

Value

(`character(1)`)
The formatted shape string.

shapevec_reprs *Format multiple shape vectors*

Description

Applies `shapevec_repr()` to multiple shapes and combines them.

Usage

```
shapevec_reprs(...)
```

Arguments

`...`	Shape vectors to format.

Value

(`character(1)`)
The formatted shapes, separated by ", ".

without *Remove elements from a vector*

Description

Remove elements from a vector by index. Also works for empty indices.

Usage

```
without(x, indices)
```

Arguments

`x`	(`vector`) The vector to remove elements from.
`indices`	(`integer()`) The indices to remove.

Value

(`vector`)
The vector with the elements removed.

Index