

Package 'lisp'

June 15, 2026

Type Package

Encoding UTF-8

Title List-Processing à La 'SRFI-1'

Version 0.2

Date 2026-06-13

Description Provides list-processing utilities inspired by the 'SRFI-1' list library for Scheme (<https://srfi.schemers.org/srfi-1/srfi-1.html>), including car/cdr family accessors, zip, pairwise, for.each, pair.fold.right and friends. Higher-order helpers that are orthogonal to list processing are deferred to the 'functional' package; this package is freely a mixture of implementation and API.

License GPL (>= 2)

URL <https://github.com/klutometis/R-lisp>

BugReports <https://github.com/klutometis/R-lisp/issues>

LazyLoad yes

Suggests RUnit

Collate 'lisp.R'

NeedsCompilation no

Author Peter Danenberg [aut, cre]

Maintainer Peter Danenberg <pcd@roxygen.org>

Repository CRAN

Date/Publication 2026-06-15 10:50:12 UTC

Contents

caar	2
cadar	3
caddr	3

cadr	4
car	4
cdddr	5
cddr	5
cdr	6
cdrs	6
for.each	7
is.even	7
is.nil	8
is.odd	8
last	9
nil	9
pair.fold.right	10
pairwise	10
zip	11
zip.c	11
zip.list	12
zip.with.names	12

Index	13
--------------	-----------

caar	<i>Composite car/cdr</i>
------	--------------------------

Description

Composite car/cdr

Usage

caar(list)

Arguments

list	the list from which to extract
------	--------------------------------

Value

The extracted elements

cadar	<i>Composite car/cdr</i>
-------	--------------------------

Description

Composite car/cdr

Usage

cadar(list)

Arguments

list the list from which to extract

Value

The extracted elements

caddr	<i>Composite car/cdr</i>
-------	--------------------------

Description

Composite car/cdr

Usage

caddr(list)

Arguments

list the list from which to extract

Value

The extracted elements

cadr	<i>Composite car/cdr</i>
------	--------------------------

Description

Composite car/cdr

Usage

cadr(list)

Arguments

list the list from which to extract

Value

The extracted elements

car	<i>First element of a list</i>
-----	--------------------------------

Description

First element of a list

Usage

car(list)

Arguments

list the list to first

Value

The first element

cdddr *Composite car/cdr*

Description

Composite car/cdr

Usage

`cdddr(list)`

Arguments

`list` the list from which to extract

Value

The extracted elements

cddr *Composite car/cdr*

Description

Composite car/cdr

Usage

`cddr(list)`

Arguments

`list` the list from which to extract

Value

The extracted elements

cdr	<i>Return elements after the first of a list.</i>
-----	---

Description

Return elements after the first of a list.

Usage

```
cdr(list)
```

Arguments

list the list from which to extract

Value

The elements after the first, or nil if only one

cdrs	<i>Try to get the cdrs; otherwise, return nil.</i>
------	--

Description

Try to get the cdrs; otherwise, return nil.

Usage

```
cdrs(...)
```

Arguments

... lists to cdr

Value

the cdr of the lists

for.each	<i>Apply f to the successive elements of</i>
----------	--

Description

Apply f to the successive elements of

Usage

```
for.each(f, . . .)
```

Arguments

f	the function to apply, whose arity should match the cardinality of . . .
. . .	lists upon which to apply f successively

Value

NULL

is.even	<i>Is a number even?</i>
---------	--------------------------

Description

Is a number even?

Usage

```
is.even(a)
```

Arguments

a	the number to test
---	--------------------

Value

Whether the number is even

is.nil *Whether a list is empty.*

Description

Whether a list is empty.

Usage

```
is.nil(list)
```

Arguments

list the list to test

Value

Whether the list is empty

is.odd *Is a number odd?*

Description

Is a number odd?

Usage

```
is.odd(a)
```

Arguments

a the number to test

Value

Whether the number is odd

last	<i>Last element in a list.</i>
------	--------------------------------

Description

Last element in a list.

Usage

```
last(list)
```

Arguments

list	The list to last
------	------------------

Value

The last element of list.

nil	<i>The empty list</i>
-----	-----------------------

Description

The empty list

Usage

```
nil
```

Format

```
list()
```

`pair.fold.right` *pair-fold-right from SRFI-1.*

Description

pair-fold-right from SRFI-1.

Usage

```
pair.fold.right(f, nil, ...)
```

Arguments

<code>f</code>	function to apply over the list-tails
<code>nil</code>	the default value
<code>...</code>	the lists whose tails fold over

Value

The result of folding `f` over the successive tails (pairs) of the input lists; `nil` if the lists are empty.

`pairwise` *Combine a list into pairwise elements; lists should be of the same length. In case of odd numbers of members, the last will be removed.*

Description

Combine a list into pairwise elements; lists should be of the same length. In case of odd numbers of members, the last will be removed.

Usage

```
pairwise(list)
```

Arguments

<code>list</code>	the list to be pairwise decomposed
-------------------	------------------------------------

Value

A list of pairwise elements

zip	<i>Zip n lists together into tuples of length n.</i>
-----	--

Description

Zip n lists together into tuples of length n .

Usage

```
zip(zipper, ...)
```

Arguments

zipper	the zipping function
...	the lists to be zipped

Value

A list of tuples

zip.c	<i>Zip using c.</i>
-------	---------------------

Description

Zip using c .

Usage

```
zip.c(...)
```

Arguments

...	the lists to be zipped
-----	------------------------

Value

A list of tuples

See Also

[zip](#)

zip.list	<i>Zip using list.</i>
----------	------------------------

Description

Zip using list.

Usage

```
zip.list(...)
```

Arguments

... the lists to be zipped

Value

A list of tuples

See Also

[zip](#)

zip.with.names	<i>Do a less efficient zip whilst preserving names.</i>
----------------	---

Description

Do a less efficient zip whilst preserving names.

Usage

```
zip.with.names(...)
```

Arguments

... lists to be zipped whilst preserving names

Value

A list of tuples (one per position) with names preserved from the input lists.

Index

* datasets

nil, 9

caar, 2

cadar, 3

caddr, 3

cadr, 4

car, 4

cddddr, 5

cddr, 5

cdr, 6

cdrs, 6

for .each, 7

is.even, 7

is.nil, 8

is.odd, 8

last, 9

nil, 9

pair.fold.right, 10

pairwise, 10

zip, 11, 11, 12

zip.c, 11

zip.list, 12

zip.with.names, 12