Package: lvec (via r-universe)

November 16, 2026

Type Package

Title Out of Memory Vectors

Version 0.2.5.9002

Author Jan van der Laan [aut, cre]

Maintainer Jan van der Laan <r@eoos.dds.nl>

Description Core functionality for working with vectors (numeric, integer, logical and character) that are too large to keep in memory. The vectors are kept (partially) on disk using memory mapping. This package contains the basic functionality for working with these memory mapped vectors (e.g. creating, indexing, ordering and sorting) and provides C++ headers which can be used by other packages to extend the functionality provided in this package. Compatibility updates by Renova.

URL https://github.com/djvanderlaan/lvec

Depends stats, R (>= 3.0.0)

LinkingTo BH, Rcpp

Imports Rcpp

Suggests testthat

License GPL-3

LazyLoad yes

RoxygenNote 7.2.1

NeedsCompilation yes

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

Date/Publication 2026-11-16 22:14:25 UTC

RemoteUrl https://github.com/renova-r/lvec

RemoteRef renova-0.2.5.9002

RemoteSha e7dffd5e0b603a2a83dba04fe7e2cef0516dde25

Contents

as_lvec . . . 2
as_rvec . . . 3
chunk . . . 3
clone . . . 4
is_lvec . . . 5
length.lvec . . . 5
lget . . . 6
lsave . . . 7
lset . . . 8
lvec . . . 9
lvec_type . . . 10
order . . . 10
print.lvec . . . 11
rattr . . . 12
sort.lvec . . . 13
strlen . . . 13

Index **15**

`as_lvec` *Converts a primitive R-vector to lvec*

Description

Converts a primitive R-vector to lvec

Usage

```
as_lvec(x)
```

Arguments

`x`	the object to convert. This can be a vector of type `character`, `integer`, `numeric` or `logical`.

Value

Returns an `lvec` of the same type as `x`. When `x` is already and `lvec`, `x` is returned. For character vectors the maximum length of the `lvec` is set to the maximum length found in `x`.

Examples

```
# convert a character vector to lvec
x <- as_lvec(letters)
lget(x, 1:26)
```

as_rvec *Convert complete lvec to R vector*

Description

Convert complete lvec to R vector

Usage

```
as_rvec(x)
```

Arguments

x	lvec to convert.

Value

Returns an R vector of type integer, numeric, character or logical.

chunk *Generate a number of index ranges from a vector*

Description

The ranges have a maximum length.

Usage

```
chunk(x, ...)

## S3 method for class 'lvec'
chunk(x, chunk_size = 1e+06, ...)

## Default S3 method:
chunk(x, chunk_size = NULL, ...)

## S3 method for class 'data.frame'
chunk(x, chunk_size = NULL, ...)
```

Arguments

x	an object for which the index ranges should be calculated. Should support the length method. For example, an lvec or a regular R vector.
...	ignored; used to pass additional arguments to other methods.
chunk_size	a numeric vector of length 1 giving the maximum length of the chunks.

Details

The default chunk size can be changes by setting the option 'chunk_size', ('options(chunk_size = <new default chunk size>)').

Implementations of chunk for data frames and regular vectors are provided to make it easier to write code that works on both lvec objects and regular R objects.

`clone` *Clone an lvec object*

Description

Clone an lvec object

Usage

```
clone(x, ...)

## S3 method for class 'lvec'
clone(x, ...)
```

Arguments

`x`	`lvec` object to clone
`...`	ignored; used to pass additional arguments to other methods

Details

`lvec` objects are basically pointers to pieces of memory. When copying an object only the pointer is copied and when modifying the copied object also the original object is modified. The advantage of this is speed: these is less copying of the complete vector. In order to obtain a true copy of an `lvec` code can be used.

Examples

```
a <- as_lvec(1:3)
# Copy
b <- a
# When modifying the copy also the original is modified
lset(b, 1, 10)
print(a)
print(b)
# Use clone to make a true copy
b <- clone(a)
lset(b, 1, 100)
print(a)
print(b)
```

is_lvec *Check if an object is of type lvec*

Description

Check if an object is of type lvec

Usage

```
is_lvec(x)
```

Arguments

x	the object to check

Value

Returns TRUE is the object is of type lvec and FALSE otherwise.

length.lvec *Get and set the length of an lvec*

Description

Get and set the length of an lvec

Usage

```
## S3 method for class 'lvec'
length(x)

## S3 replacement method for class 'lvec'
length(x) <- value
```

Arguments

x	the lvec
value	the new length of the link{lvec}

Value

The length of the lvec.

lget *Read elements from an lvec*

Description

Read elements from an lvec

Usage

```
lget(x, ...)

## S3 method for class 'lvec'
lget(x, index = NULL, range = NULL, ...)

## Default S3 method:
lget(x, index = NULL, range = NULL, ...)

## S3 method for class 'data.frame'
lget(x, index = NULL, range = NULL, ...)
```

Arguments

x	the lvec to read from
...	used to pass on additional arguments to other methods.
index	a logical or numeric vector to index x with
range	a numeric vector of length 2 specifying a range of elements to select. Specify either index or range.

Details

Indexing using index should follow the same rules as indexing a regular R-vector using a logical or numeric index. The range given by range includes both end elements. So, a range of c(1,3) selects the first three elements.

Value

Returns an lvec with the selected elements. In order to convert the selection to an R-vector as_rvec can e used.

Examples

```
a <- as_lvec(letters[1:4])
# Select first two elements
lget(a, 1:2)
lget(a, c(TRUE, TRUE, FALSE, FALSE))
lget(a, range = c(1,2))
```

```
# Logical indices are recycled: select odd elements
lget(a, c(TRUE, FALSE))
```

`lsave` *Read and write lvec object to file*

Description

Read and write lvec object to file

Usage

```
lsave(x, filename, overwrite = TRUE, compress = FALSE)

lload(filename)
```

Arguments

`x`	`lvec` object to save
`filename`	name of the file(s) to save the lvec to. See details.
`overwrite`	overwrite existing files or abort when files would be overwritten.
`compress`	a logical specifying if the data should be compressed. (see `saveRDS`).

Details

The `lvec` is written in chunks to a number of RDS files using `saveRDS`. When filename contains the extension 'RDS' (capitalisation may differ), this extension is stripped from the filename. After that the lvec is written in blocks (or chunks) to files having names `<filename>.00001.RDS`, `<filename>.00002.RDS` etc. Some additional data (data type, the number of blocks, the size of the lvec, etc) is written to the file `<filename>.RDS`.

The size of the chunks can be controlled by the option 'chunk_size' (see `chunk`).

Value

`lsave` does not return anything. `lload` returns an `lvec`.

`lset` *Set values in an lvec*

Description

Set values in an lvec

Usage

```
lset(x, ...)

## S3 method for class 'lvec'
lset(x, index = NULL, values, range = NULL, ...)

## Default S3 method:
lset(x, index = NULL, values, range = NULL, ...)

## S3 method for class 'data.frame'
lset(x, index = NULL, values, range = NULL, ...)
```

Arguments

`x`	`lvec` to set values in
`...`	used to pass additional arguments to other methods
`index`	a numeric or logical vector with indices at which the values should be set.
`values`	a vector with the new values. When shorter than the length of the indices the values are recycled.
`range`	a numeric vector of length 2 specifying a range of elements to select. Specify either `index` or `range`.

Details

Should behave in the same way as assigning and indexing to a regular R-vector. The range given by `range` includes both end elements. So, a range of `c(1,3)` selects the first three elements.

When `range` is given, and `values` is not given it is assumed `index` contains the values. Therefore, one can do `lset(x, range = c(1,4), NA)`, to set the first four elements of `x` to missing.

Examples

```
a <- as_lvec(1:10)
# set second element to 20
lset(a, 2, 20)
print(a)
# set odd elements to 20
lset(a, c(TRUE, FALSE), 20)
print(a)
# values are recycled
```

```
lset(a, 1:4, 100:101)
print(a)
# range index; set first 3 elements to NA
lset(a, range = c(1,3), NA)
print(a)
```

`lvec` *Create memory mapped vector*

Description

The data in these vectors are stored on disk (partially buffered for speed) allowing one to work with more data than fits into available memory.

Usage

```
lvec(
  size,
  type = c("numeric", "integer", "logical", "character"),
  strlen = NULL
)
```

Arguments

`size`	the size of the vector
`type`	the type of the vector. Should be one of the following value: `"numeric"`, `"integer"`, `"logical"` or `"character"`. The types will create vectors corresponding to the corresponding R types.
`strlen`	in case of a vector of type `"character"` the maximum length of the strings should also be specified using `strlen`.

Details

The minimum value of `strlen` is two. When a value smaller than that is given it is automatically set to two. This is because a minimum of two bytes is necessary to also store missing values correctly.

Value

Returns an object of type `lvec`. Elements of this vector are stored on file (partially buffered in memory for speed) allowing one to work with more data than fits into memory.

Examples

```
# create an integer vector of length 100
x <- lvec(100, type = "integer")
# Get the first 10 values; values are initialised to 0 by default
lget(x, 1:10)
# Set the first 10 values to 11:20
lset(x, 1:10, 11:20)

# set maximum length of the string to 1, strings longer than that get
# truncated. However, minimum value of strlen is 2.
x <- lvec(10, type = "character", strlen = 1)
lset(x, 1:3, c("a", "foo", NA))
lget(x, 1:3)
```

`lvec_type` *Get the type of the lvec*

Description

Get the type of the lvec

Usage

```
lvec_type(x)
```

Arguments

`x`	the `lvec` to get the type from.

Value

Returns an character vector of length 1 with one of the following values: `"integer"`, `"numeric"`, `"logical"` or `"character"`.

`order` *Order a lvec*

Description

Order a lvec

Usage

```
order(x, ...)

## Default S3 method:
order(x, ...)

## S3 method for class 'lvec'
order(x, ...)
```

Arguments

x	lvec to sort
...	unused.

Value

Returns the order of x. Unlike the default order function in R, the sort used is not stable (e.g. in case there are multiple records with the same value in x, there relative order after sorting is not defined).

Examples

```
x <- as_lvec(rnorm(10))
order(x)
```

---

print.lvec *Print an lvec*

Description

Print an lvec

Usage

```
## S3 method for class 'lvec'
print(x, ...)
```

Arguments

x	lvec to print.
...	unused

Value

Returns x invisibly.

rattr *Set and get attributes of the original R-vector stored in an lvec*

Description

Set and get attributes of the original R-vector stored in an lvec

Usage

```
rattr(x, which)

rattr(x, which) <- value
```

Arguments

x	and object of type lvec
which	a character vector of length one giving the name of the attribute.
value	the new value of the attribute

Details

The attributes of the lvec can be set and obtained using the standard functions attr and attributes. However when an lvec is converted to an R-vector using, for example, as_rvec, the attributes of the resulting R-vector are set using the result of rattr. This can be used to store vectors such as factors and dates (POSIXct) in lvec objects, as these are basically integer and numeric vectors with a number of additional attributes.

Examples

```
dates <- as_lvec(as.Date("2016-12-05", "2016-12-24"))
# When printing and reading the result is converted back to a date object
print(dates)
as_rvec(dates)

# make a factor of an integer lvec
a <- as_lvec(1:3)
rattr(a, "class") <- "factor"
rattr(a, "levels") <- c("a", "b", "c")
print(a)
```

sort.lvec *Sort a lvec*

Description

Sort a lvec

Usage

```
## S3 method for class 'lvec'
sort(x, decreasing = FALSE, clone = TRUE, ...)
```

Arguments

x	lvec to sort
decreasing	unused (a value unequal to FALSE will generate an error).
clone	clone x before sorting
...	unused.

Value

Sorts x and returns a sorted copy of x. When clone is FALSE the input vector is modified.

Examples

```
x <- as_lvec(rnorm(10))
sort(x)

# Effect of clone
a <- as_lvec(rnorm(10))
b <- sort(a, clone = FALSE)
print(a)
```

strlen *Get and set the maximum string length of a character lvec*

Description

Get and set the maximum string length of a character lvec

Usage

```
strlen(x)

strlen(x) <- value
```

Arguments

x	a lvec of type character.
value	the new value of the maximum string length.

Examples

```
a <- as_lvec('123')
strlen(a)  # = 3
# Strings are truncated to strlen
lset(a, 1, '123456')
print(a)  # '123'
strlen(a) <- 5
lset(a, 1, '123456')
print(a)  # '12345'
```

Index

as_lvec, 2
as_rvec, 3, *6*, *12*
attr, *12*
attributes, *12*

chunk, 3, *7*
clone, 4

is_lvec, 5

length, *3*
length.lvec, 5
length<-.lvec *(*length.lvec*)*, 5
lget, 6
lload *(*lsave*)*, 7
lsave, 7
lset, 8
lvec, *2–8*, 9, *10–12*
lvec_type, 10

order, 10, *11*

print.lvec, 11

rattr, 12
rattr<- *(*rattr*)*, 12

saveRDS, *7*
sort.lvec, 13
strlen, 13
strlen<- *(*strlen*)*, 13