Package: sparsevctrs (via r-universe)

June 29, 2024

Title Sparse Vectors for Use in Data Frames

Version 0.1.0.9000

Description Provides sparse vectors powered by ALTREP (Alternative Representations for R Objects) that behave like regular vectors, and can thus be used in data frames. Also provides tools to convert between sparse matrices and data frames with sparse columns and functions to interact with sparse vectors.

License MIT + file LICENSE

URL https://github.com/r-lib/sparsevctrs, https://r-lib.github.io/sparsevctrs/

BugReports https://github.com/r-lib/sparsevctrs/issues

Depends R (>= 4.0.0)

Imports cli (>= 3.4.0), rlang (>= 1.1.0), vctrs

Suggests knitr, Matrix, methods, rmarkdown, testthat (>= 3.0.0), tibble, withr

VignetteBuilder knitr

Config/Needs/website tidyverse/tidytemplate, rmarkdown, lobstr, ggplot2, bench, tidyr, ggbeeswarm

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1.9000

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

RemoteUrl https://github.com/r-lib/sparsevctrs

RemoteRef HEAD

RemoteSha 9c22ca9a1b153f8a83ea3d6b7260a089b1d8bf8d

Contents

`coerce-vector` *Coerce numeric vector to sparse double*

Description

Takes a numeric vector, integer or double, and turn it into a sparse double vector.

Usage

```
as_sparse_double(x, default = 0)

as_sparse_integer(x, default = 0L)

as_sparse_character(x, default = "")

as_sparse_logical(x, default = FALSE)
```

Arguments

`x`	a numeric vector.
`default`	default value to use. Defaults to `0`.
	The values of `x` must be double or integer. It must not contain any `Inf` or `NaN` values.

Value

sparse vectors

Examples

```
x_dense <- c(3, 0, 2, 0, 0, 0, 4, 0, 0, 0)
x_sparse <- as_sparse_double(x_dense)
x_sparse

is_sparse_double(x_sparse)
```

`coerce_to_sparse_data_frame` *Coerce sparse matrix to data frame with sparse columns*

Description

Turning a sparse matrix into a data frame

Usage

```
coerce_to_sparse_data_frame(x)
```

Arguments

x	sparse matrix.

Details

The only requirement from the sparse matrix is that it contains column names.

Value

data.frame with sparse columns

See Also

`coerce_to_sparse_tibble()` `coerce_to_sparse_matrix()`

Examples

```
set.seed(1234)
mat <- matrix(sample(0:1, 100, TRUE, c(0.9, 0.1)), nrow = 10)
colnames(mat) <- letters[1:10]
sparse_mat <- Matrix::Matrix(mat, sparse = TRUE)
sparse_mat

res <- coerce_to_sparse_data_frame(sparse_mat)
res

# All columns are sparse
vapply(res, is_sparse_vector, logical(1))
```

coerce_to_sparse_matrix *Coerce sparse data frame to sparse matrix*

Description

Turning data frame with sparse columns into sparse matrix using `Matrix::sparseMatrix()`.

Usage

```
coerce_to_sparse_matrix(x)
```

Arguments

x	a data frame or tibble with sparse columns.

Details

No checking is currently do to `x` to determine whether it contains sparse columns or not. Thus it works with any data frame. Needless to say, creating a sparse matrix out of a dense data frame is not ideal.

Value

sparse matrix

See Also

`coerce_to_sparse_data_frame()` `coerce_to_sparse_tibble()`

Examples

```
sparse_tbl <- lapply(1:10, function(x) sparse_double(x, x, length = 10))
names(sparse_tbl) <- letters[1:10]
sparse_tbl <- as.data.frame(sparse_tbl)
sparse_tbl

res <- coerce_to_sparse_matrix(sparse_tbl)
res
```

coerce_to_sparse_tibble *Coerce sparse matrix to tibble with sparse columns*

Description

Turning a sparse matrix into a tibble.

Usage

```
coerce_to_sparse_tibble(x)
```

Arguments

x	sparse matrix.

Details

The only requirement from the sparse matrix is that it contains column names.

Value

tibble with sparse columns

See Also

`coerce_to_sparse_data_frame()` `coerce_to_sparse_matrix()`

Examples

```
set.seed(1234)
mat <- matrix(sample(0:1, 100, TRUE, c(0.9, 0.1)), nrow = 10)
colnames(mat) <- letters[1:10]
sparse_mat <- Matrix::Matrix(mat, sparse = TRUE)
sparse_mat

res <- coerce_to_sparse_tibble(sparse_mat)
res

# All columns are sparse
vapply(res, is_sparse_vector, logical(1))
```

`extractors` *Information extraction from sparse vectors*

Description

Extract positions, values, and default from sparse vectors without the need to materialize vector.

Usage

```
sparse_positions(x)

sparse_values(x)

sparse_default(x)
```

Arguments

x	vector to be extracted from.

Details

`sparse_default()` returns `NA` when applied to non-sparse vectors. This is done to have an indicator of non-sparsity.

for ease of use, these functions also works on non-sparse variables.

Value

vectors of requested attributes

Examples

```
x_sparse <- sparse_double(c(pi, 5, 0.1), c(2, 5, 10), 10)
x_dense <- c(0, pi, 0, 0, 0.5, 0, 0, 0, 0, 0.1)

sparse_positions(x_sparse)
sparse_values(x_sparse)
sparse_default(x_sparse)

sparse_positions(x_dense)
sparse_values(x_dense)
sparse_default(x_dense)

x_sparse_3 <- sparse_double(c(pi, 5, 0.1), c(2, 5, 10), 10, default = 3)
sparse_default(x_sparse_3)
```

`sparse_character` *Create sparse character vector*

Description

Construction of vectors where only values and positions are recorded. The Length and default values determine all other information.

Usage

```
sparse_character(values, positions, length, default = "")
```

Arguments

`values`	integer vector, values of non-zero entries.
`positions`	integer vector, indices of non-zero entries.
`length`	integer value, Length of vector.
`default`	integer value, value at indices not specified by `positions`. Defaults to `""`. Cannot be `NA`.

Details

`values` and `positions` are expected to be the same length, and are allowed to both have zero length.

Allowed values for `value` are character values. Missing values such as `NA` and `NA_real_` are allowed as they are turned into `NA_character_`. Everything else is disallowed. The values are also not allowed to take the same value as `default`.

`positions` should be integers or integer-like doubles. Everything else is not allowed. Positions should furthermore be positive (`0` not allowed), unique, and in increasing order. Lastly they should all be smaller that `length`.

For developers:

setting `options("sparsevctrs.verbose_materialize" = TRUE)` will print a message each time a sparse vector has been forced to materialize.

Value

sparse character vector

See Also

`sparse_double()` `sparse_integer()`

Examples

```
sparse_character(character(), integer(), 10)

sparse_character(c("A", "C", "E"), c(2, 5, 10), 10)

str(
  sparse_character(c("A", "C", "E"), c(2, 5, 10), 1000000000)
)
```

`sparse_double` *Create sparse double vector*

Description

Construction of vectors where only values and positions are recorded. The Length and default values determine all other information.

Usage

```
sparse_double(values, positions, length, default = 0)
```

Arguments

`values`	double vector, values of non-zero entries.
`positions`	integer vector, indices of non-zero entries.
`length`	integer value, Length of vector.
`default`	double value, value at indices not specified by `positions`. Defaults to `0`. Cannot be `NA`.

Details

`values` and `positions` are expected to be the same length, and are allowed to both have zero length.

Allowed values for `value` is double and integer values. integer values will be coerced to doubles. Missing values such as `NA` and `NA_real_` are allowed. Everything else is disallowed, This includes `Inf` and `NaN`. The values are also not allowed to take the same value as `default`.

`positions` should be integers or integer-like doubles. Everything else is not allowed. Positions should furthermore be positive (`0` not allowed), unique, and in increasing order. Lastly they should all be smaller that `length`.

For developers:

setting `options("sparsevctrs.verbose_materialize" = TRUE)` will print a message each time a sparse vector has been forced to materialize.

Value

sparse double vector

See Also

sparse_integer() sparse_character()

Examples

```
sparse_double(numeric(), integer(), 10)

sparse_double(c(pi, 5, 0.1), c(2, 5, 10), 10)

str(
  sparse_double(c(pi, 5, 0.1), c(2, 5, 10), 1000000000)
)
```

---

sparse_integer *Create sparse integer vector*

Description

Construction of vectors where only values and positions are recorded. The Length and default values determine all other information.

Usage

```
sparse_integer(values, positions, length, default = 0L)
```

Arguments

values	integer vector, values of non-zero entries.
positions	integer vector, indices of non-zero entries.
length	integer value, Length of vector.
default	integer value, value at indices not specified by positions. Defaults to 0L. Cannot be NA.

Details

values and positions are expected to be the same length, and are allowed to both have zero length.

Allowed values for value is integer values. This means that the double vector c(1, 5, 4) is accepted as it can be losslessly converted to the integer vector c(1L, 5L, 4L). Missing values such as NA and NA_real_ are allowed. Everything else is disallowed, This includes Inf and NaN. The values are also not allowed to take the same value as default.

positions should be integers or integer-like doubles. Everything else is not allowed. Positions should furthermore be positive (0 not allowed), unique, and in increasing order. Lastly they should all be smaller that length.

For developers:

setting options("sparsevctrs.verbose_materialize" = TRUE) will print a message each time a sparse vector has been forced to materialize.

Value

sparse integer vector

See Also

`sparse_double()` `sparse_character()`

Examples

```
sparse_integer(integer(), integer(), 10)

sparse_integer(c(4, 5, 7), c(2, 5, 10), 10)

str(
  sparse_integer(c(4, 5, 7), c(2, 5, 10), 1000000000)
)
```

`sparse_logical` *Create sparse logical vector*

Description

Construction of vectors where only values and positions are recorded. The Length and default values determine all other information.

Usage

```
sparse_logical(values, positions, length, default = FALSE)
```

Arguments

`values`	logical vector, values of non-zero entries.
`positions`	integer vector, indices of non-zero entries.
`length`	integer value, Length of vector.
`default`	logical value, value at indices not specified by `positions`. Defaults to `FALSE`. Cannot be `NA`.

Details

`values` and `positions` are expected to be the same length, and are allowed to both have zero length.

Allowed values for `value` are logical values. Missing values such as `NA` and `NA_real_` are allowed. Everything else is disallowed, The values are also not allowed to take the same value as `default`.

`positions` should be integers or integer-like doubles. Everything else is not allowed. Positions should furthermore be positive (`0` not allowed), unique, and in increasing order. Lastly they should all be smaller that `length`.

For developers:

setting `options("sparsevctrs.verbose_materialize" = TRUE)` will print a message each time a sparse vector has been forced to materialize.

Value

sparse logical vector

See Also

sparse_double() sparse_integer() sparse_character()

Examples

```
sparse_logical(logical(), integer(), 10)

sparse_logical(c(TRUE, NA, TRUE), c(2, 5, 10), 10)

str(
  sparse_logical(c(TRUE, NA, TRUE), c(2, 5, 10), 1000000000)
)
```

---

type-predicates *Sparse vector type checkers*

Description

Helper functions to determine whether an vector is a sparse vector or not.

Usage

```
is_sparse_vector(x)

is_sparse_numeric(x)

is_sparse_double(x)

is_sparse_integer(x)

is_sparse_character(x)

is_sparse_logical(x)
```

Arguments

x	value to be checked.

Details

`is_sparse_vector()` is a general function that detects any type of sparse vector created with this package. `is_sparse_double()`, `is_sparse_integer()`, `is_sparse_character()`, and `is_sparse_logical()` are more specific functions that only detects the type. `is_sparse_numeric()` matches both sparse integers and doubles.

Value

single logical value

Examples

```
x_sparse <- sparse_double(c(pi, 5, 0.1), c(2, 5, 10), 10)
x_dense <- c(0, pi, 0, 0, 0.5, 0, 0, 0, 0, 0.1)

is_sparse_vector(x_sparse)
is_sparse_vector(x_dense)

is_sparse_double(x_sparse)
is_sparse_double(x_dense)

is_sparse_character(x_sparse)
is_sparse_character(x_dense)

# Forced materialization
is_sparse_vector(x_sparse[])
```

Index