Package 'smvr'

June 27, 2025

Title Simple Implementation of Semantic Versioning

Version 0.1.0

Description Simple implementation of Semantic Versioning 2.0.0 on the 'vctrs' package. This package provides a simple way to create, compare, and manipulate semantic versions in R. It is designed to be lightweight and easy to use.

License MIT + file LICENSE

URL https://eitsupi.github.io/smvr/, https://github.com/eitsupi/smvr

BugReports https://github.com/eitsupi/smvr/issues

Depends R (>= 4.1)

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

Suggests dplyr, testthat (>= 3.0.0), tibble

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Tatsuya Shima [aut, cre]

Maintainer Tatsuya Shima <ts1s1andn@gmail.com>

Repository CRAN

Date/Publication 2025-06-27 13:40:02 UTC

Contents

`as_smvr` *Convert to* `smvr` *vector*

Description

`as_smvr()` is a generic function that converts an object to `smvr` vector. The default method uses `vctrs::vec_cast()` to convert the object.

Usage

```
as_smvr(x, ...)

## Default S3 method:
as_smvr(x, ...)
```

Arguments

`x`	An object to convert to `smvr`.
`...`	Additional arguments passed to methods.

Value

An object of class `smvr`.

Examples

```
as_smvr(c("1.0.0", "2.0.0-rc.1", "3.0.0+build.1"))
as_smvr(numeric_version(c("1", "2.3")))
as_smvr(NA)
```

`check-component` *Check if the* `smvr` *object has a specific component*

Description

These functions check if the smvr object has a specific component.

- `is_pre_release()`: Checks if the pre-release identifiers are present.
- `has_build_metadata()`: Checks if the build metadata is present.

Usage

```
is_pre_release(x)

has_build_metadata(x)
```

Arguments

x	A smvr object.

Value

Indicates whether x has the specified component.

See Also

- extract-component functions for extracting components from a smvr object.

Examples

```
v <- parse_semver(c(
  "1.0.0", "2.0.0-alpha", "2.0.0-beta", "2.0.0-beta.2+build.123"
))
v

is_pre_release(v)
has_build_metadata(v)
```

---

extract-component *Extract each component of version numbers/labels*

Description

These functions extract the individual components of version numbers or labels, such as major, minor, patch numbers, or, pre-release identifiers and build metadata.

Usage

```
extract_major(x, ...)

extract_minor(x, ...)

extract_patch(x, ...)

extract_pre_release_ids(x, ...)

extract_build_metadata(x, ...)

## S3 method for class 'smvr'
extract_major(x, ...)

## S3 method for class 'smvr'
extract_minor(x, ...)

## S3 method for class 'smvr'
```

```
extract_patch(x, ...)

## S3 method for class 'smvr'
extract_pre_release_ids(x, ...)

## S3 method for class 'smvr'
extract_build_metadata(x, ...)

## S3 method for class 'numeric_version'
extract_major(x, ...)

## S3 method for class 'numeric_version'
extract_minor(x, ...)

## S3 method for class 'numeric_version'
extract_patch(x, ...)
```

Arguments

x	A version object.
...	Additional arguments passed to methods.

Value

The extracted component of the version object.

- `extract_major()`, `extract_minor()`, and `extract_patch()` return integer.
- `extract_pre_release_ids()` returns pre_release_ids.
- `extract_build_metadata()` returns character.

See Also

- check-component functions for checking if smvr object has a specific component.

Examples

```
sem_ver <- parse_semver(c("1.2.3-alpha+001", "2.0.0", NA))

extract_major(sem_ver)
extract_minor(sem_ver)
extract_patch(sem_ver)
extract_pre_release_ids(sem_ver)
extract_build_metadata(sem_ver)

# Extracting version also works for numeric_version
num_ver <- numeric_version(c("1", "3.1.4.1.5", NA), strict = FALSE)

extract_major(num_ver)
extract_minor(num_ver)
extract_patch(num_ver)
```

is_smvr *Check if an object is a* smvr *object*

Description

Check if an object is a smvr object

Usage

```
is_smvr(x)
```

Arguments

x	An object.

Value

Indicates whether x is a smvr object.

Examples

```
is_smvr(smvr(1, 2, 3))
```

new_pre_release_identifier *Single pre-release identifier*

Description

A class representing a single pre-release identifier (alphanumeric or numeric) for Semantic Versioning 2.0.0.

Usage

```
new_pre_release_identifier(x = character())
```

Arguments

x	Something that can be coerced to a character vector by vctrs::vec_cast(). Empty string ("") is a special case that means no identifier. The default is length 0 character.

Value

A pre_release_identifier vector.

See Also

- pre_release_ids: Whole pre-release identifiers (Concatenation of pre_release_identifier).

Examples

```
ids <- new_pre_release_identifier(
  c("1", "2", "10", "01", "alpha", "beta", "", NA)
)
ids

# Numeric identifiers are always sorted before alphanumeric ones.
vctrs::vec_sort(ids)

# Works with base R vectors.
ids[ids == "alpha" & !is.na(ids)]
ids[ids > 2L & !is.na(ids)]
```

---

new_pre_release_ids *Pre-release identifiers*

Description

Create a vector of pre-release identifiers, which can be used for marking versions as pre-release versions.

- `pre_release_ids()` is a low-level constructor for creating pre-release identifiers from individual components.
- `parse_pre_release_ids()` parses a character vector into pre-release identifiers.

Empty identifiers are special cases that indicate *not a pre-release version*.

Usage

```
new_pre_release_ids(id1 = character(), id2 = "", id3 = "", id4 = "", id5 = "")

parse_pre_release_ids(x)
```

Arguments

`id1`, `id2`, `id3`, `id4`, `id5`	Single pre-release identifiers. Each identifier can be something to be cast to a pre_release_identifier vector by `vctrs::vec_cast()`.
`x`	A character vector representing pre-release identifiers. Each identifier separated by a dot (`"."`) will be parsed as a pre_release_identifier.

Details

Internally, pre_release_ids store up to 5 pre_release_identifier. So this can't represent pre-release identifiers with more than 5 components. If passing character containing more than 5 components to `parse_pre_release_ids()`, it will throw an error.

Value

A pre_release_ids vector.

Examples

```
# Each components are concatenated with a dot
new_pre_release_ids("rc", 1:3)

ids <- parse_pre_release_ids(
  c("", "alpha.beta", "alpha.1", "beta", "beta.11", "beta.2")
)
ids

# Empty ids have the highest precedence
# (Used to indicate not a pre-release version)
vctrs::vec_sort(ids)

# Works with base R vectors.
ids[ids > "beta.2"]
```

`parse_semver` *A vector representing versions following Semantic Versioning*

Description

The `smvr` class represents versions that follow the Semantic Versioning Specification (SemVer).

- `smvr()` is a constructor for creating `smvr` objects from each component.
- `parse_semver()` parses a character vector into `smvr` objects.

Usage

```
parse_semver(x)

smvr(major = integer(), minor = 0L, patch = 0L, pre_release = "", build = "")
```

Arguments

`x`	A character vector representing semantic versions. Each version should follow the Semantic Versioning Specification. Partial matches are not allowed (e.g., `"1.0"` is not valid).
`major`, `minor`, `patch`	Non-negative integers representing the major, minor, and patch version components. The default values for `minor` and `patch` are `0`.
`pre_release`	Something that can be cast to a pre_release_ids vector. This represents pre-release identifiers, which can be empty (`""`) meaning non pre-release.
`build`	Optional build metadata character vector. Should have the pattern `^[a-zA-Z0-9-]+` and can contain multiple components separated by dots (`"."`). This can be empty (`""`) meaning no build metadata.

Details

Build metadata is not used for ordering, but the == and != operators check it and exactly same build metadata is required for equality. The other operators (<, <=, >, >=) ignore build metadata.

Value

A smvr class vector.

Examples

```
# SemVer versions from components
smvr(4, 1:5)

# Parse SemVer versions from character
parse_semver(c("1.0.0-alpha", "1.0.0-beta+exp.sha.5114f85"))

v <- parse_semver(c(
  "1.0.0",
  "1.0.0-alpha",
  "1.0.0-beta",
  "1.0.0-rc.1",
  "1.0.0-rc.2",
  NA
))
v

# Sorting
vctrs::vec_sort(v)

# Works with base R vectors.
v[v >= "1.0.0-rc.2"]

# Partial version components are treated as NA
suppressWarnings(parse_semver("1.5"))

# The numeric_version class supports versions with
# less than 3 components, and can be cast to smvr.
numeric_version("1.5") |>
  vctrs::vec_cast(smvr())
```

---

update-version *Update version components*

Description

These functions allows to update the components of version objects.

- `increment_major()`, `increment_minor()`, and `increment_patch()` update the major, minor, and patch version numbers respectively.

- `mark_as_pre_release()` marks the version as a pre-release version.
- `add_build_metadata()` adds build metadata to the version.

Usage

```
increment_major(x, ...)

## S3 method for class 'smvr'
increment_major(x, ...)

increment_minor(x, ...)

## S3 method for class 'smvr'
increment_minor(x, ...)

increment_patch(x, ...)

## S3 method for class 'smvr'
increment_patch(x, ...)

mark_as_pre_release(x, ...)

## S3 method for class 'smvr'
mark_as_pre_release(x, ids = "pre", ...)

add_build_metadata(x, ...)

## S3 method for class 'smvr'
add_build_metadata(x, metadata = "", ...)
```

Arguments

`x`	An version object
`...`	Additional arguments passed to methods.
`ids`	A character vector of pre-release identifiers.
`metadata`	A character vector of build metadata.

Value

An updated version object with the specified changes applied.

Examples

```
v <- parse_semver(c("0.9.9", "1.0.0-a.1", "1.1.0+1"))

increment_major(v)
increment_minor(v)
increment_patch(v)
mark_as_pre_release(v, ids = c("pre.1"))
```

```
add_build_metadata(v, metadata = "build.1")
```

Index