Package 'smvr'

May 9, 2026

Title Simple Implementation of Semantic Versioning (SemVer)

Version 0.2.2

Description Simple implementation of Semantic Versioning 2.0.0 ('SemVer') 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 2025-10-13 14:10:02 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

A smvr class vector.

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

- `smvr()` to create a smvr object from components.
- 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_ids` *Pre-release identifiers*

Description

A class representing a collection of identifiers, which are used for representing pre-release versions.

There are two functions to create the pre_release_ids vector:

- `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.

Usage

```
new_pre_release_ids(...)

parse_pre_release_ids(x)
```

Arguments

...	<dynamic-dots> Single pre-release identifiers. Each identifier can be something to be cast to a pre_release_identifier vector by vctrs::vec_cast(). All components must be of the same length or length 1 (will be recycled).
x	A character vector representing pre-release identifiers. Each identifier separated by a dot (.) will be parsed as a pre_release_identifier.

Details

If the components are empty, they are treated as the highest precedence pre-release ids, which is used to indicate that the version is *not a pre-release version*.

Value

A pre_release_ids vector.

Limitations

There are some limitations on the number of identifiers in some operations:

- When comparing with a string, the number of identifiers in the string. If it exceeds 5, an error is raised.
- When assigning, the number of identifiers in the value being assigned. If it exceeds the number of identifiers in the target or 5, whichever is larger, an error is raised.

Please refer to the examples for details.

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)

# Can be compared with string notation
ids[ids > "beta.2"]

# Limitations:
# 1. When comparing with a string, the number of identifiers in the string
#    must not exceed 5.
try(ids[ids > "beta.2.3.4.5.6"])

# This works since the string is parsed first.
ids[ids > parse_pre_release_ids("beta.2.3.4.5.6")]
```

```
# 2. When assigning, the number of identifiers in the value being assigned
#    must not exceed the number of identifiers in the target or 5,
#    whichever is larger.
try(ids[1] <- parse_pre_release_ids("beta.2.3.4.5.6"))
```

`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()`. Each element must be ASCII alphanumerics, hyphens, or empty string (`""`). Empty string is a special case that means no identifier.

Details

Identifiers are compared based on the following criteria:

- If the identifier is empty, it is treated as the smallest value.
- Integers greater than or equal to 0 are treated as numeric identifiers and compared numerically.
- Else, identifiers are treated as alphanumeric identifiers and compared lexically ASCII sort order.
- Numeric identifiers always have lower precedence than alphanumeric identifiers.

Value

A pre_release_identifier vector.

See Also

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

Examples

```
id <- new_pre_release_identifier(
  c("1", "2", "10", "0a", "-1", "alpha", "beta", "", NA)
)
id

# empty < numeric < alphanumeric
vctrs::vec_sort(id)

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

SEM_VER_PATTERN *A suggested regular expression (RegEx) to check a SemVer string*

Description

This is a suggested regular expression (RegEx) to check a SemVer string, without the "^" at the start and "$" at the end. It is useful to extract the SemVer components from VCS tags etc.

Usage

```
SEM_VER_PATTERN
```

Format

An object of class `character` of length 1.

Value

Single string

Examples

```
SEM_VER_PATTERN

# VCS tag names often have a "v" prefix to SemVer
tag_names <- c("v1.0.0", "v1.1.0-alpha.1", "v1.1.0+build.1", "not-a-version")

# Extract and parse SemVer
regmatches(tag_names, m = regexpr(SEM_VER_PATTERN, tag_names)) |>
  parse_semver()
```

smvr *A vector representing versions following Semantic Versioning*

Description

The smvr class represents versions that follow the Semantic Versioning Specification (SemVer). A version number contains three components, MAJOR.MINOR.PATCH, and optional pre-release and build metadata labels.

This is similar to the base R's numeric_version class, but always has three components (major, minor, patch) and supports pre-release and build metadata labels. And, unlike numeric_version, SemVer uses dots (.) as separators and does not allow hyphens (-) except to indicate the start of a pre-release label.

There are two functions to create smvr objects:

- `smvr()` is a constructor from each component. Each component must have the same length or length 1 (will be recycled).
- `parse_semver()` parses a character vector.

Usage

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

parse_semver(x)
```

Arguments

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 can be empty ("") meaning non pre-release (default).
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 (default).
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).

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.

See Also

- as_smvr() to convert other classes to smvr.
- extract-component functions to extract components of a smvr object. (Operations opposite to smvr()).
- update-version functions to update components of a smvr object.

Examples

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

# Parse SemVer versions from character
v <- parse_semver(c(
  "1.0.0",
  "1.0.0-alpha",
  "1.0.0-beta",
  "1.0.0-rc.2",
  "1.0.0-rc.10",
  NA
))
v

# Sorting
vctrs::vec_sort(v)

# Can be compared with string notation
v[v >= "1.0.0-rc.2" & !is.na(v)]

# 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())

# Be careful with hyphens in numeric_version and SemVer.
# The following examples yield opposite results.
numeric_version("1.0.0-1") > "1.0.0" # 1.0.0-1 is the same as 1.0.0.1
parse_semver("1.0.0-1") > "1.0.0"    # 1.0.0-1 is a pre-release version
```

---

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. Note that these functions reset the pre-release and build metadata to empty.
- `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, ...)

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`	Something can be cast to pre_release_ids representing the pre-release identifiers, length must be 1 or the same as `x`.
`metadata`	A character vector of build metadata, length must be 1 or the same as `x`.

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

Index