Package: vecvec (via r-universe)

June 3, 2026

Title Construct Mixed Type Data Structures with Vectors of Vectors

Version 1.1.0

Description Mixed type vectors are useful for combining semantically similar classes. Some examples of semantically related classes include time across different granularities (e.g. daily, monthly, annual) and probability distributions (e.g. Normal, Uniform, Poisson). These groups of vector types typically share common statistical operations which vary in results with the attributes of each vector. The 'vecvec' data structure facilitates efficient storage and computation across multiple vectors within the same object.

License MIT + file LICENSE

Imports rlang, S7, stats, vctrs

Suggests lifecycle, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL https://pkg.mitchelloharawild.com/vecvec/, https://github.com/mitchelloharawild/vecvec

BugReports https://github.com/mitchelloharawild/vecvec/issues

Config/roxygen2/version 7.3.3.9000

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

Date/Publication 2026-06-03 19:17:09 UTC

RemoteUrl https://github.com/mitchelloharawild/vecvec

RemoteRef HEAD

RemoteSha a35de8d160ee09cfae92881a0dbccad35d3caeb8

Contents

`class_vecvec` *S7 class for vecvec*

Description

`class_vecvec()` constructs a new vecvec object from a list of vectors. It is designed to be performant with minimal checks on the inputs. Direct use of `class_vecvec()` is useful for developers, but for users it is recommended to use `vecvec()` to create vecvec objects.

Usage

```
class_vecvec(x = list(), i = seq_len(sum(lengths(x))))
```

Arguments

`x`	An unnamed list of arbitrary vectors. The vectors can be of different types and lengths, but they must all be vectors (according to vctrs::vec_is). Adjacent vectors of the same type will be combined together into a single vector.
`i`	A vector of integers specifying the location of each element in `x` as if they were combined in order. The values in `i` must be between 1 and the total number of elements across all vectors in `x`, and can contain duplicates. If not provided, it defaults to a sequence from 1 to the total number of elements across all vectors in `x`.

Value

A vector of vectors of S7 object of class `class_vecvec`.

Examples

```
# Create a vecvec prototype
class_vecvec()

# Construct a vecvec from a list of vectors
class_vecvec(list(letters, rnorm(10)))
```

```
# Fully specify a vecvec with locations
class_vecvec(
  x = list(letters, rnorm(10)),
  i = c(1:3, 27:31, 26:4, 32:36)
)
```

`is_vecvec` *Test if an object is a vecvec*

Description

Test if an object is a vecvec

Usage

```
is_vecvec(x)
```

Arguments

`x`	Object to test.

Value

`TRUE` if `x` inherits from `class_vecvec`, `FALSE` otherwise.

See Also

`vecvec()` to create a vecvec object.

Examples

```
vv <- vecvec(1:3, letters)
is_vecvec(vv)
is_vecvec(1:3)
```

masked-cor *Correlation, Variance and Covariance*

Description

var, cov, and cor compute the variance of x and the covariance or correlation of x and y if these are vectors. If x and y are matrices then the covariances (or correlations) between the columns of x and the columns of y are computed.

These functions mask and wrap the stats::var, stats::cov, and stats::cor functions to add support for vecvec objects. More details can be found in the documentation for those functions.

Usage

```
var(x, ...)

cov(x, ...)

cor(x, ...)
```

Arguments

x	a numeric vector, matrix or data frame.
...	Additional arguments passed to stats::var, stats::cov, and stats::cor.

See Also

stats::var(), stats::cov(), stats::cor()

unvecvec *Coerce a vector of vectors to a single typed vector*

Description

[Stable]

Flattens a vecvec into a plain R vector by casting all elements to a common type. This is the inverse of vecvec().

Type resolution follows vctrs coercion rules: when ptype is NULL the common type is determined automatically from the slots of x via vctrs::vec_ptype_common(). If no common type can be found (e.g. all slots are empty), the result falls back to logical().

Usage

```
unvecvec(x, ptype = NULL)
```

Arguments

x	A vecvec object.
ptype	A prototype specifying the desired output type, e.g. character() or numeric(). If NULL (the default), the common type is inferred from the elements of x using vctrs::vec_ptype_common(), falling back to logical() when no common type can be determined. Passing an explicit ptype is useful when you need a guaranteed output type regardless of what x contains, or when automatic inference would pick an undesirable type.

Value

A vector of length length(x) and type ptype (or the inferred common type when ptype = NULL). Positions corresponding to NA indices in the underlying vecvec structure are filled with NA.

See Also

vecvec() to create a vecvec; vctrs::vec_ptype_common() for the type inference rules; vctrs::vec_cast() for the casting rules.

Examples

```
vv <- vecvec(1:3, c(4.5, 5.5))

# Automatic type inference: integer + double -> double
unvecvec(vv)

# Force a specific output type
unvecvec(vv, ptype = character())
```

---

vecvec *Create a vector of vectors*

Description

[Stable]

A vecvec is a vector that holds elements of different types without coercing them to a common type. Unlike a list of vectors, a vecvec behaves as a flat vector (hence vector of vectors). This means that you can operations (such as indexing, arithmetic, and statistics) apply across all elements of a vecvec as if they were combined into a single vector.

Mixed-type vectors can be useful if you need to store heterogeneous data in a vector-like structure. In most cases, I believe this is bad practice for data analytics, but this could be useful for tidying up messy data. The most valuable use case for vecvec is as a data structure for mixed-type semantic vectors. This package is used by mixtime and distributional to create vectors of time with different chronons and distributions with different shapes.

To convert a vecvec back to a plain typed vector, use unvecvec(), which casts all elements to a common type via vctrs::vec_cast().

Usage

```
vecvec(...)
```

Arguments

`...`	Vectors to combine. Each vector is stored as a separate typed slot; no type coercion is performed.

Value

A `vecvec` object whose length equals the total number of elements across all input vectors.

See Also

`unvecvec()` to coerce a `vecvec` to a single-typed vector; `is_vecvec()` to test whether an object is a `vecvec`.

Examples

```
# Mixed types are preserved without coercion
vv <- vecvec(Sys.Date(), rnorm(3), letters)
vv

# .[i] Indexing works like a flat vector
vv[c(1L, 3L, 7L)]

# .[[i]] drops to the original vector type
vv[[2L]]
```

`vecvec_apply` *Apply a function to each vector of the vecvec*

Description

The `vecvec_apply()` function applies a function `.f` to each vector in the `vecvec` vectors.

Usage

```
vecvec_apply(x, .f, ...)
```

Arguments

`x`	A vecvec object
`.f`	A function to apply to each vector
`...`	Additional arguments passed to `.f`

Value

A vecvec data type with the same structure as `x` but with each vector transformed by `.f`.

`vecvec_apply_fn` *Function factory for vecvec_apply*

Description

The `vecvec_apply_fn()` function is a function factory that creates applies the function `.f` to each vector and optionally simplifies the result with `[unvecvec()]`. The function matches the forms of the original function `.f` and can be used to define methods for generic functions that apply to `vecvec` objects. If `.f` is a primitive function, the resulting function will have a apply over an argument `x` and pass through `...`.

Usage

```
vecvec_apply_fn(.f, ptype = NULL, SIMPLIFY = !is.null(ptype))
```

Arguments

`.f`	A function to apply to each vector
`ptype`	A prototype to simplify to. If `NULL`, the result will be a `vecvec` object. If not `NULL`, the result will be simplified to the type of `ptype` if possible.
`SIMPLIFY`	If `TRUE`, the `[unvecvec()]` will be applied to the result of `vecvec_apply()`, using `ptype` as the target type. If `FALSE`, the result will always be a `vecvec` object.

Value

A function that applies `.f` to each vector of a `vecvec` object and optionally simplifies the result.

`vecvec_register` *Register methods for a vecvec subclass*

Description

Call `vecvec_register()` inside your package's `.onLoad()` function to register the S3 methods that allow a custom vecvec subclass to participate in the vctrs vector system.

Usage

```
vecvec_register(x)
```

Arguments

x	An S7 class object that extends class_vecvec. Typically the class object created by `S7::new_class()` in your package, e.g. `class_myvec`.

Details

Specifically, `vecvec_register()` registers:

- `vec_cast.<class>.*` methods, so that other types can be cast *to* your vecvec subclass.
- `vec_cast.*.<class>` methods, so that your vecvec subclass can be cast *from* other types.

Value

`NULL`, invisibly. Called for its side effects.

Examples

```
## Not run:
# In your package, define a vecvec subclass:
class_myvec <- S7::new_class(
  "myvec",
  parent = vecvec::class_vecvec
)

# Then register it in .onLoad():
.onLoad <- function(libname, pkgname) {
  S7::methods_register()
  vecvec::vecvec_register(class_myvec)
}

## End(Not run)
```

Index