Package 'term'

June 20, 2020

Title Create, Manipulate and Query Parameter Terms

Version 0.2.0

Description Creates, manipulates, queries and repairs vectors
of parameter terms. Parameter terms are the labels used to reference
values in vectors, matrices and arrays. They represent the names in
coefficient tables and the column names in 'mcmc' and 'mcmc.list'
objects.

License MIT + file LICENSE

URL `https://github.com/poissonconsulting/term`

BugReports `https://github.com/poissonconsulting/term/issues`

Depends R (>= 3.4)

Imports chk, extras (>= 0.0.1), lifecycle, rlang, universals, vctrs

Suggests covr, testthat

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.0.9000

NeedsCompilation no

Author Joe Thorley [aut, cre] (<https://orcid.org/0000-0002-7683-4592>),
Kirill Müller [aut] (<https://orcid.org/0000-0002-1416-3412>),
Ayla Pearson [ctb] (<https://orcid.org/0000-0001-7388-1222>),
Poisson Consulting [cph, fnd]

Maintainer Joe Thorley <`joe@poissonconsulting.ca`>

Repository CRAN

Date/Publication 2020-06-20 09:20:09 UTC

R topics documented:

`as_term` *Coerce to a Term Vector*

Description

Coerces an R object to a `term-vector()`.

Usage

```
as_term(x, ...)

as.term(x, ...)

## S3 method for class 'character'
as_term(x, repair = FALSE, ...)
```

```
## S3 method for class 'numeric'
as_term(x, name = "par", ...)
```

Arguments

`x`	The object.
`...`	Unused.
`repair`	A flag specifying whether to repair terms.
`name`	A string specifying the name of the parameter.

Details

`as.term` has been **Soft-deprecated** for `as_term`.

Methods (by class)

- `character`: Coerce character vector to term vector
- `numeric`: Coerce default object to term vector

See Also

`term-vector()` and `repair_terms()`

Examples

```
as_term(matrix(1:4, 2))
as_term(c("parm3[10]", "parm3[2]", "parm[2,2]", "parm[1,1]"))
```

chk_term *Check Term*

Description

Checks if term using `vld_term()`.

Usage

```
chk_term(x, validate = "complete", x_name = NULL)
```

Arguments

`x`	The object.
`validate`	A string specifying the level of the validation. The possible values in order of increasing strictness are 'class', 'valid', 'consistent' and 'complete'.
`x_name`	A string of the name of object x or NULL.

Value

`NULL`, invisibly. Called for the side effect of throwing an error if the condition is not met.

Examples

```
x <- term("x[2]", "x[1]")
chk_term(x)
x <- c("x[2]", "x[1]")
try(chk_term(x, validate = "sorted"))
```

`complete_terms` *Complete Terms*

Description

Completes an object's terms.

Usage

```
complete_terms(x, ...)

## S3 method for class 'term'
complete_terms(x, ...)
```

Arguments

`x`	The object.
`...`	Unused.

Details

The term vector is repaired before being completed. Missing values are ignored.

Methods (by class)

- `term`: Complete Terms for a term Vector

See Also

`term-vector()`, `repair_terms()` and `is_incomplete_terms()`.

Examples

```
complete_terms(term("b[3]", "b[1]", "b[2]"))
complete_terms(term("z[2,2]", "z[1,1]"))
```

`consistent_term` *Consistent Terms*

Description

Test whether the number of dimensions of terms in the same parameter are consistent.

Usage

```
consistent_term(x)
```

Arguments

`x`	The object.

Value

A logical vector indicating whether the number of dimensions is consistent.

See Also

`term-vector()` and `npdims()`

Examples

```
consistent_term(term("alpha[1]", "alpha[3]", "beta[1,1]", "beta[2,1]"))
consistent_term(term("alpha[1]", NA_term_, "beta[1,1]", "beta[2]"))
```

`dims.term` *Dimensions*

Description

Gets the dimensions of an object.

Usage

```
## S3 method for class 'term'
dims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Details

Unlike `base::dim()`, dims returns the length of an atomic vector.

Value

An integer vector of the dimensions.

See Also

`base::dim()`

Other dimensions: `ndims()`, `npdims()`, `pdims()`

Examples

```
dims(term("beta[1,1]"))
dims(term("beta[1,1]", "beta[1,2]"))
```

`is_incomplete_terms` *Is Incomplete Terms*

Description

Tests whether a term vector has absent elements. The vector should not require repairing.

Usage

```
is_incomplete_terms(x, ...)
```

Arguments

`x`	The object.
`...`	Unused.

Value

A logical scalar indicating whether the object's terms are incomplete.

See Also

`term-vector()` and `complete_terms()`

Examples

```
is_incomplete_terms(term("b[2]"))
is_incomplete_terms(term("b[2]", "b[1]"))
is_incomplete_terms(term("b[2]", "b[1]", "b[1]"))
```

`is_inconsistent_terms` *Is Inconsistent Terms*

Description

Tests whether a term vector has inconsistent elements.

Usage

```
is_inconsistent_terms(x, ...)
```

Arguments

`x`	The object.
`...`	Unused.

Value

A logical scalar indicating whether the object's terms are inconsistent.

See Also

`term-vector()` and `consistent_term()`

Examples

```
is_inconsistent_terms(term("b[2]"))
is_inconsistent_terms(term("b[2]", "b[1]"))
is_inconsistent_terms(term("b[2]", "b[1,1]"))
```

`is_term` *Is Term*

Description

Tests whether an R object inherits from S3 class `term`.

Usage

```
is_term(x)
```

Arguments

`x`	The object.

Details

It does not test the validity of consistency of the term elements.

Value

A flag indicating whether the test was positive.

See Also

`term-vector()`, `vld_term()`, `valid_term()` and `consistent_term()`

Examples

```
is_term(c("parameter[2]", "parameter[10]"))
is_term(term("parameter[2]", "parameter[10]"))
```

`NA_term_` *Missing Term*

Description

A missing term element.

Usage

```
NA_term_
```

Format

An object of class `term` (inherits from `vctrs_vctr`) of length 1.

See Also

`term-vector()`

Examples

```
is_term(NA_term_)
is.na(NA_term_)
```

`new_term` *Construct a new term object*

Description

Use this function to quickly construct a term object from a character vector, without checking the input. Use `term()` to repair the input.

Usage

```
new_term(x = character())
```

Arguments

`x`	A character vector.

Examples

```
new_term()
new_term(c("a", "b[1]", "b[2]"))

# Terms are not checked for validity:
new_term("r[")
repair_terms(new_term("r["))
```

`npars.term` *Number of Parameters*

Description

Gets the number of parameters of an object.

The default methods returns the length of `pars()` if none are `NA`, otherwise it returns `NA`.

Usage

```
## S3 method for class 'term'
npars(x, scalar = NULL, ...)
```

Arguments

`x`	An object.
`scalar`	A flag specifying whether to by default return all parameters (NULL), or only scalar parameters (TRUE) or only non-scalar parameters (FALSE).
`...`	Other arguments passed to methods.

Value

An integer scalar of the number of parameters.

See Also

pars()

Other MCMC dimensions: nchains(), niters(), nsams(), nsims(), nterms()

Other parameters: pars(), set_pars()

Examples

```
npars(term("sigma", "alpha[1]", "alpha[2]", "beta[1,1]", "beta[2,1]"))
```

---

npdims.term *Number of Dimensions of each Parameter*

Description

Gets the number of parameters of an object.

The default methods returns the length of pars() if none are NA, otherwise it returns NA.

Usage

```
## S3 method for class 'term'
npdims(x, terms = FALSE, ...)
```

Arguments

x	An object.
terms	A flag specifying whether to get the number of dimensions for each term element.
...	Other arguments passed to methods.

Value

An integer scalar of the number of parameters.

See Also

pars()

Other MCMC dimensions: nchains(), niters(), nsams(), nsims(), nterms()

Other parameters: pars(), set_pars()

Examples

```
npdims(term("alpha[1]", "alpha[3]", "beta[1,1]", "beta[2,1]"))
```

`nterms.term` *Number of Terms of a term*

Description

Gets the number of unique terms.

Usage

```
## S3 method for class 'term'
nterms(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Details

If the term vector includes missing values then it returns NA_integer.

Value

A integer scalar of the number of terms.

See Also

Other MCMC dimensions: `nchains()`, `niters()`, `npars()`, `nsams()`, `nsims()`

Examples

```
nterms(term("alpha[1]", "alpha[2]", "beta[1,1]", "beta[2,1]"))
nterms(term("alpha[1]", "alpha[1]", "beta[1,1]", "beta[1,1]"))
```

`pars.character` *Parameter Names*

Description

Gets the parameter names.

Usage

```
## S3 method for class 'character'
pars(x, scalar = NULL, ...)
```

Arguments

`x`	An object.
`scalar`	A flag specifying whether to by default return all parameters (NULL), or only scalar parameters (TRUE) or only non-scalar parameters (FALSE).
`...`	Other arguments passed to methods.

Value

A character vector of the names of the parameters.

See Also

Other parameters: `npars()`, `set_pars()`

Examples

```
pars(c("a", "b[1]", "a[3]"))
```

`pars.default` *Parameter Names*

Description

Gets the parameter names.

Usage

```
## Default S3 method:
pars(x, scalar = NULL, ...)
```

Arguments

`x`	An object.
`scalar`	A flag specifying whether to by default return all parameters (NULL), or only scalar parameters (TRUE) or only non-scalar parameters (FALSE).
`...`	Other arguments passed to methods.

Value

A character vector of the names of the parameters.

See Also

Other parameters: `npars()`, `set_pars()`

Examples

```
pars.foobar <- function(x, ...) {
  NotYetImplemented()
  # replace with code to get pars for an object of class 'foobar'
}
```

---

pars.term *Parameter Names*

Description

Gets the parameter names.

Usage

```
## S3 method for class 'term'
pars(x, scalar = NULL, terms = FALSE, ...)
```

Arguments

x	An object.
scalar	A flag specifying whether to by default return all parameters (NULL), or only scalar parameters (TRUE) or only non-scalar parameters (FALSE).
terms	A flag specifying whether to return the parameter name for each term element.
...	Other arguments passed to methods.

Value

A character vector of the names of the parameters.

See Also

Other parameters: pars_terms()

Examples

```
term <- term(
  "alpha[1]", "alpha[2]", "beta[1,1]", "beta[2,1]",
  "beta[1,2]", "beta[2,2]", "sigma", NA
)
pars(term)
pars(term, scalar = TRUE)
pars(term, scalar = FALSE)
```

`pars_terms` *Term Parameters*

Description

Gets the name of each parameter for each term.

Usage

```
pars_terms(x, scalar = NULL, ...)
```

Arguments

`x`	A term vector.
`scalar`	A flag specifying whether to by default return all parameters (NULL), or only scalar parameters (TRUE) or only non-scalar parameters (FALSE).
`...`	Unused.

Details

The scalar argument has been **Soft-deprecated**.

Value

A character vector of the term parameter names.

See Also

Other parameters: `pars.term()`

Examples

```
term <- term(
  "alpha[1]", "alpha[2]", "beta[1,1]", "beta[2,1]",
  "beta[1,2]", "beta[2,2]", "sigma", NA
)
pars_terms(term)
```

`pdims.term` *Parameter Dimensions*

Description

Gets the dimensions of each parameter of an object.

Usage

```
## S3 method for class 'term'
pdims(x, ...)
```

Arguments

`x`	An object.
`...`	Other arguments passed to methods.

Details

Errors if the parameter dimensions are inconsistent.

A named list of the dimensions of each parameter can be converted into the equivalent `term-vector()` using `term()`.

Value

A named list of integer vectors of the dimensions of each parameter.

See Also

Other dimensions: `dims()`, `ndims()`, `npdims()`

Examples

```
pdims(term("alpha[1]", "alpha[3]", "beta[1,1]", "beta[2,1]"))
```

`repair_terms` *Repair Terms*

Description

Repairs a terms vector.

Usage

```
repair_terms(x)
```

Arguments

`x`	The object.

Details

Invalid elements are replaced by missing values and spaces removed.

If a parameter such as `b` is a scalar then `b[1]` is replaced by `b` but if higher indices are included such as `b[2]` then `b` is replaced by `b[1]`. Note that repairing does not necessarily establish consistency or completeness, see `vld_term()` for details.

Value

The repaired term vector.

See Also

`term-vector()` and `valid_term()`

Examples

```
repair_terms(new_term(c("b[3]", "b")))
repair_terms(new_term(c("a[3]", "b[1]")))
repair_terms(new_term(c("a [3]", " b [ 1  ] ")))
repair_terms(new_term(c("a", NA)))
```

`set_pars.term` *Set Parameter Names*

Description

Sets an object's parameter names.

The assignment version pars<-() forwards to `set_pars()`.

Usage

```
## S3 method for class 'term'
set_pars(x, value, ...)
```

Arguments

`x`	An object.
`value`	A character vector of the new parameter names.
`...`	Other arguments passed to methods.

Details

`value` must be a unique character vector of the same length as the object's parameters.

Value

The modified object.

See Also

Other parameters: `npars()`, `pars()`

Examples

```
set_pars.foobar <- function(x, ...) {
  NotYetImplemented()
  # replace with code to set_pars for an object of class 'foobar'
}
```

`subset.term` *Subset Term Vector*

Description

Subsets a term vector.

Usage

```
## S3 method for class 'term'
subset(x, pars = NULL, select = NULL, ...)
```

Arguments

`x`	The object.
`pars`	A character vector of parameter names.
`select`	A character vector of the names of the parameters to include in the subsetted object.
`...`	Unused.

Details

The `select` argument has been **Soft-deprecated** for `pars`.

Value

The modified term vector.

See Also

`term-vector()`

Examples

```
term <- term(
  "alpha[1]", "alpha[2]", "beta[1,1]", "beta[2,1]",
  "beta[1,2]", "beta[2,2]", "sigma"
)
subset(term, "beta")
subset(term, c("alpha", "sigma"))
```

---

term *Term Vector*

Description

Creates a term vector from values. A `term` vector is an S3 vector of parameter terms of the form p, q[#] or r[#,#] where # are positive integers. This function checks that all terms are valid but does not require stronger levels of consistency, see `chk_valid()` for details.

Usage

```
term(...)
```

Arguments

`...`	Unnamed values are term values, named values describe the parameter in the name and the dimensionality in the value.

Value

A term vector.

See Also

`dims()` and `pdims()`

Examples

```
term()
term("p", "q[1]", "q[2]", "q[3]")
term("q[1]", "q[2]", "q[3]")
combined <- term(par = 2:4, "alpha")
pdims(combined)
term(!!!pdims(combined))

# Invalid terms are rejected:
try(term("r["))

# Valid terms are repaired
term("r  [ 1  ,2  ]")
```

`tindex` *Term Index*

Description

Gets the index for each term of an object.

Usage

```
tindex(x)
```

Arguments

`x`	The object.

Details

For example the index of `beta[2,1]` is `c(2L,1L)` while the index for `sigma` is `1L`. It is useful for extracting the values of individual terms.

Value

A named list of integer vectors of the index for each term.

Examples

```
tindex(term("alpha", "alpha[2]", "beta[1,1]", "beta[2 ,1  ]"))
```

`valid_term` *Valid Terms*

Description

Test whether each element in a term vector is valid.

Usage

```
valid_term(x)
```

Arguments

`x`	The object.

Details

Repairing a term vector replaces invalid terms with a missing value.

Value

A logical vector indicating whether each term is valid.

See Also

`term-vector()` and `repair_terms()`

Examples

```
# valid term elements
valid_term(term("a", "a [3]", " b [ 1  ] ", "c[1,300,10]"))
# invalid term elements
valid_term(new_term(c("a b", "a[1]b", "a[0]", "b[1,]", "c[]", "d[1][2]")))
```

`vec_cast.term` *Cast a vector to a specified type*

Description

`vec_cast()` provides directional conversions from one type of vector to another. Along with `vec_ptype2()`, this generic forms the foundation of type coercions in vctrs.

Usage

```
## S3 method for class 'term'
vec_cast(x, to, ...)
```

Arguments

`x`	Vectors to cast.
`to`	Type to cast to. If `NULL`, `x` will be returned as is.
`...`	For `vec_cast_common()`, vectors to cast. For `vec_cast()`, `vec_cast_default()`, and `vec_restore()`, these dots are only for future extensions and should be empty.

See Also

`vctrs::vec_cast()`

Examples

```
vec_cast(new_term(c("a[1]", "a[2]")), character())
vec_cast(c("a[1]", "a[2]"), new_term())
```

`vec_ptype2.term` *Find the common type for a pair of vectors*

Description

`vec_ptype2()` defines the coercion hierarchy for a set of related vector types. Along with `vec_cast()`, this generic forms the foundation of type coercions in vctrs.

`vec_ptype2()` is relevant when you are implementing vctrs methods for your class, but it should not usually be called directly. If you need to find the common type of a set of inputs, call `vec_ptype_common()` instead. This function supports multiple inputs and finalises the common type.

Usage

```
## S3 method for class 'term'
vec_ptype2(x, y, ...)
```

Arguments

`x`	Vector types.
`y`	Vector types.
`...`	These dots are for future extensions and must be empty.

See Also

`vctrs::vec_ptype2()`

`vld_term` *Validate Term*

Description

Validates the elements of a term vector. Use `chk_s3_class()` to check if an object is a term.

Usage

```
vld_term(x, validate = "complete")
```

Arguments

`x`	The object.
`validate`	A string specifying the level of the validation. The possible values in order of increasing strictness are 'class', 'valid', 'consistent' and 'complete'.

Details

Internal validity of a term can be checked on three levels:

- `"valid"` checks that all terms are of the form `x`, x[#], x[#,#] etc. where `x` is an identifier and # are positive integers.
- `"consistent"` checks that all terms are addressed with the same dimensionality; the terms `x[1]` and `x[2,3]` are inconsistent.
- `"complete"` checks that the values span all possible values across all dimensions; if `x[3,4]` exist, the vector must contain at least 11 more terms to be consistent (`x[1,1]` to `x[1,4]`, `x[2,1]` to `x[2,4]` and `x[3,1]` to `x[3,3]`).

Missing values are ignored as are duplicates and order.

Value

A flag indicating whether the condition was met.

See Also

`chk_term()`

Examples

```
vld_term(c("x[2]", "x[1]"))
vld_term(term("x[2]", "x[1]"))
```

Index