Package ‘term’

January 15, 2020

Title  Create, Manipulate and Query Parameter Terms
Version  0.1.0
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
Depends  R (>= 3.3)
Imports  chk, lifecycle
Suggests  covr, testthat
URL  https://github.com/poissonconsulting/term
BugReports  https://github.com/poissonconsulting/term/issues
RdMacros  lifecycle
Encoding  UTF-8
LazyData  true
RoxygenNote  7.0.2
Language  en-US
NeedsCompilation  no
Author  Joe Thorley [aut, cre] (<https://orcid.org/0000-0002-7683-4592>), Ayla Pearson [ctb] (<https://orcid.org/0000-0001-7388-1222>), Poisson Consulting [cph, fnd]
Maintainer  Joe Thorley <joe@poissonconsulting.ca>
Repository  CRAN
Date/Publication  2020-01-15 22:30:02 UTC

R topics documented:

  as.character.term .................................................. 2
c.term ................................................................. 3
as.character.term

Coerce to a Term Vector

Description

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

Usage

```r
## S3 method for class 'term'
as.character(x, ...)

as.term(x, ...)

## S3 method for class 'character'
as.term(x, repair = FALSE, ...)

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

Arguments

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

Methods (by class)

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

See Also
term-vector() and repair_terms()

Examples

```r
as.term(matrix(1:4, 2))
as.term(c("parm3[10]", "parm3[2]", "parm[2,2]", "parm[1,1]"))
```

c.term Combine Term Vectors

Description

Combine Term Vectors

Usage

```r
## S3 method for class 'term'
c(..., recursive = FALSE)
```

Arguments

... Term vectors to concatenate
recursive Ignored.

Value

A term vector.

See Also
term-vector()

Examples

```r
c(NA_term_, "b", "a[1]", "")
```
chk_term  
**Check Term**

**Description**
Checks if term using \texttt{vld_term}(x).

**Usage**
\begin{verbatim}
chk_term(x, validate = "class", x_name = NULL)
\end{verbatim}

**Arguments**
- \texttt{x}  
The object to check.
- \texttt{validate}  
  A string specifying the level of the validation. The possible values in order of increasing strictness are 'class', 'valid', 'consistent' and 'complete'.
- \texttt{x_name}  
  A string of the name of object \texttt{x} or NULL.

**Value**
\texttt{NULL}, invisibly. Called for the side effect of throwing an error if the condition is not met.

**Examples**
\begin{verbatim}
x <- as.term(c("x[2]", "x[1]"))
chk_term(x)
x <- c("x[2]", "x[1]")
try(chk_term(x, validate = "sorted"))
\end{verbatim}

calculate_terms  
**Complete Terms**

**Description**
Add any absent elements to a terms vector.

**Usage**
\begin{verbatim}
calculate_terms(x, ...)
\end{verbatim}

**Arguments**
- \texttt{x}  
The object.
- \texttt{...}  
  Unused.
consistent_term

Details

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

Value

The repaired and complete term vector.

Methods (by class)

- `term`: Complete terms of a terms vector

See Also

`term-vector()`, `repair_terms()` and `is.incomplete_terms()`.

Examples

```r
complete_terms(as.term(c("b[3]", "b[1]", "b[2]")))
complete_terms(as.term(c("z[2,2]", "z[1,1]")))
```
**Description**

Gets the dimensions of an object.

**Usage**

```r
dims(x, ...)  
```

## Default S3 method:

dims(x, ...)

## S3 method for class 'data.frame'

dims(x, ...)

**Arguments**

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

**Details**

Unlike `base::dim()`, `dims` works as intuitively expected for vectors and `data.frame`. A integer vector of the dimensions of a parameter can be converted into the equivalent `term-vector()` using `term()`.

**Value**

An integer vector of the dimensions.

**Methods (by class)**

- default: Dimensions of a default object
- `data.frame`: Dimensions of a data frame

**See Also**

`base::dim()`, `term-vector()` and `term()`

**Examples**

```r
dims(character(0))
dims(1:3)
dims(data.frame())
dims(data.frame(x = 1:3))
```
is.incomplete_terms  Is Incomplete Terms

Description

Tests whether a term vector has absent elements.

Usage

is.incomplete_terms(x, ...)

## S3 method for class 'term'
is.incomplete_terms(x, ...)

Arguments

x  The object.

...  Unused.

Value

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

Methods (by class)

• term: Test whether a term vector is incomplete

See Also

term-vector() and complete_terms()

Examples

is.incomplete_terms(as.term("b[2]"))
is.incomplete_terms(as.term(c("b[2]", "b[1]%22")))
is.inconsistent_terms  

**Description**

Tests whether a term vector has inconsistent elements.

**Usage**

```r
is.inconsistent_terms(x, ...)  
## S3 method for class 'term'
is.inconsistent_terms(x, ...)
```

**Arguments**

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

**Value**

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

**Methods (by class)**

- `term`: Test whether a term vector is inconsistent

**See Also**

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

**Examples**

```r
is.inconsistent_terms(as.term("b[2]"))
is.inconsistent_terms(as.term(c("b[2]", "b[1]")))
is.inconsistent_terms(as.term(c("b[2]", "b[1,1]")))
```
**is.term**

*Is Term*

---

**Description**

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

**Usage**

```r
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()`, `valid_term()` and `consistent_term()`

**Examples**

```r
is.term(c("parameter[2]", "parameter[10]"))
is.term(as.term(c("parameter[2]", "parameter[10]")))
```

---

**NA_term_**

*Missing Term*

---

**Description**

A missing term element.

**Usage**

```r
NA_term_
```

**Format**

An object of class `term` (inherits from `character`) of length 1.
See Also
term-vector()

Examples

is.term(NA_term_)
is.na(NA_term_)

ndims

Number of Dimensions of an Object

Description

Gets the number of dimensions of an object as defined by dims().

Usage

ndims(x, ...)

## Default S3 method:
ndims(x, ...)

## S3 method for class 'matrix'
ndims(x, ...)

## S3 method for class 'data.frame'
ndims(x, ...)

Arguments

x The object.
...

Value

A count of the number of dimensions.

Methods (by class)

- default: Number of dimensions of a default object
- matrix: Number of dimensions of a matrix
  Always 2L.
- data.frame: Number of dimensions of a data.frame
  Always 2L.
npars

See Also
dims()

Examples
ndims(character(0))
ndims(1:3)
ndims(data.frame())
ndims(data.frame(x = 1:3))

npars | Number of Parameters
---|---

Description
Gets the number of parameters of an object as returned by pars().

Usage
npars(x, ...)

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

Arguments
x The object.
... Unused.
scalar A flag specifying whether to only count parameters with one term.

Value
A count of the number of parameters.

Methods (by class)
• term: Number of parameters of term vector

See Also
pars() and term-vector()

Examples
npars(as.term(c("sigma", "alpha[1]", "alpha[2]", "beta[1,1]", "beta[2,1]")))
npars(as.term(c("sigma", "alpha[1]", "alpha[2]")), scalar = TRUE)
npdims | Number of Parameter Dimensions

Description

Gets the number of parameter dimensions of an object as returned by `pdims()`.

Usage

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

Arguments

- `x` The object.
- `...` Unused.
- `terms` A flag specifying whether to get the number of dimensions for each term element.

Value

A named integer vector of the number of dimensions of each parameter.

Methods (by class)

- `term`: Number of parameter dimensions of term vector

See Also

`pdims()` and `term-vector()`

Examples

```r
npdims(as.term(c("alpha[1]", "alpha[3]", "beta[1,1]", "beta[2,1]")))
npdims(as.term(c("alpha[1]", "alpha[3]", "beta[1,1]", "beta[2,1]")), terms = TRUE)
```
### nterms

#### Description

Gets the number of unique term elements of an object.

#### Usage

```r
nterms(x, ...)
```

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

#### Arguments

- **x**: The object.
- **...**: Unused.

#### Details

By default if the vector includes missing values then it returns `NA_integer_`.

#### Value

A count of the number of terms.

#### Methods (by class)

- **term**: Number of unique term elements of term vector

#### See Also

- `term-vector()`

#### Examples

```r
nterms(as.term(c("alpha[1]", "alpha[2]", "beta[1,1]", "beta[2,1]")))
nterms(as.term(c("alpha[1]", "alpha[1]", "beta[1,1]", "beta[1,1]")))
```
### Description

Gets or sets the parameter names for an object.

### Usage

- `parameters(x, ...)`
- `parameters(x) <- value`
- `set_parameters(x, pars)`
- `pars(x, ...)`
- `pars(x) <- value`
- `set_pars(x, pars)`
- `## S3 method for class 'term'
pars(x, scalar = NA, terms = FALSE, ...)`

### Arguments

- **x**: The object.
- **...**: Unused.
- **value**: A character vector of the new parameter names.
- **pars**: A character vector of the new parameter names.
- **scalar**: A logical scalar specifying whether to get the names of all parameters (NA), only scalars (TRUE) or all parameters except scalars (FALSE).
- **terms**: A flag specifying whether to return the parameter name for each term element.

### Details

- `parameters()` is deprecated.
- `parameters<-` is deprecated.
- `set_parameters()` is deprecated.

### Value

A character vector of the parameter names.

### Methods (by class)

- **term**: Parameter names for a term vector
Examples

```r
term <- as.term(c("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, terms = TRUE)
```

---

params

## Parameter Descriptions for `term` Functions

### Description

Parameter Descriptions for term Functions

### Arguments

- **x**: The object.
- **name**: A string specifying the name of the parameter.
- **...**: Unused.
- **repair**: A flag specifying whether to repair terms.
- **select**: A character vector of the names of the parameters to include in the subsetted object.
- **value**: A character vector of the new parameter names.
- **pars**: A character vector of the new parameter names.
- **incomparables**: Ignored.

---

pdims

## Parameter Dimensions

### Description

Gets the parameter dimensions of an object as a named list of the dimensions of each parameter.

### Usage

```r
pdims(x, ...)
```

### Arguments

- **x**: The object.
- **...**: Unused.
Details

Errors if the parameter dimensions are inconsistent.
A named list of the dimensions of each parameter can be converted into the equivalent \texttt{term-vector()} using \texttt{term()}.

Value

A named list of the dimensions of each parameter.

See Also

dims() and term()

Examples

\begin{verbatim}
pdims(as.term(c("alpha[1]", "alpha[3]", "beta[1,1]", "beta[2,1]")))
\end{verbatim}

repair_terms

\textit{Repair Terms}

Description

Repairs a terms vector.

Usage

\texttt{repair_terms(x)}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} The object.
\end{itemize}

Details

Invalid elements are replaced by missing values and spaces removed.
If a parameter such as \texttt{b} is a scalar then \texttt{b[1]} is replaced by \texttt{b} but if higher indices are included such as \texttt{b[2]} then \texttt{b} is replaced by \texttt{b[1]}.

Value

The repaired term vector.

See Also

term-vector() and valid_term()
Examples

```r
repair_terms(as.term(c("b[3]", "b")))
repair_terms(as.term(c("a[3]", "b[1]")))
repair_terms(as.term(c("a [3]", " b [ 1 ] ")))
repair_terms(as.term(c("a", NA)))
```

### subset.term

**Subset Term Vector**

**Description**

Subsets a term vector.

**Usage**

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

**Arguments**

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

**Value**

The modified term vector.

**See Also**

`term-vector()`

**Examples**

```r
term <- as.term(c(
  "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"))
```
Description

Gets the index for each term of an object.

Usage

tdims(x)
tindex(x, ...)

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

Arguments

x The object.
...

Unused.

Details

tdims() is deprecated.

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 the index for each term.

Methods (by class)

- term: Term indices for term vector

See Also
term-vector()

Examples

tindex(as.term(c("alpha", "alpha[2]", "beta[1,1]", "beta[2,1]")))
Description

Creates a term vector from object dimensions.

Usage

term(x, ...)

## S3 method for class 'integer'
term(x, name = "par", ...)

Arguments

x                    An integer vector of the object dimensions or a named list of the parameter's dimensions.
...                   Unused.
name                  A string specifying the name of the parameter.

Value

A term vector.

Methods (by class)

- integer: Term vector from an integer vector of the object's dimensions

See Also

term-vector(), dims() and pdims()

Examples

term(0L)
term(1L)
term(2L)
## term-object  

**Term Object**

**Description**

A term object is a S3 vector of parameter terms.

**Examples**

```r
as.term(c("a[1]", "a[2]"))
as.term(matrix(1:4, 2))
term(2L)
term(c(2L, 2L), "alpha")
```

## unique.term  

**Extract Unique Terms**

**Description**

Extracts unique elements from a term vector.

**Usage**

```r
## S3 method for class 'term'
unique(x, incomparables = FALSE, ...)
```

**Arguments**

- `x`  The object.
- `incomparables`  Ignored.
- `...`  Unused.

**Value**

A vector of unique terms.

**See Also**

- `term-vector()`

**Examples**

```r
unique(as.term(c("b", "a", "a[1]", "a", NA, "aa", NA)))
```
**valid_term**

<table>
<thead>
<tr>
<th>valid_term</th>
<th>Valid Terms</th>
</tr>
</thead>
</table>

**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**

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

---

**vld_term**

<table>
<thead>
<tr>
<th>vld_term</th>
<th>Validate Term</th>
</tr>
</thead>
</table>

**Description**

Validates the elements of a term vector.

**Usage**

`vld_term(x, validate = "class")`
Arguments

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

Details

Missing values are ignored as are duplicates and order.

Value

A flag indicating whether the condition was met.

See Also

- `chk_term()`

Examples

```r
vld_term(c("x[2]", "x[1]
```
Index

*Topic datasets
  - NA_term_, 9
  - as.character.term, 2
  - as.term(as.character.term), 2
  - base::dim(), 6
  - c.term, 3
  - chk_term, 4
  - chk_term(), 22
  - complete_terms, 4
  - complete_terms(), 7
  - consistent_term, 5
  - consistent_term(), 8, 9
  - dims, 6
  - dims(), 10, 11, 16, 19
  - is.incomplete_terms, 7
  - is.incomplete_terms(), 5
  - is.inconsistent_terms, 8
  - is.term, 9
  - NA_term_, 9
  - ndims, 10
  - npars, 11
  - npdims, 12
  - npdims(), 5
  - nterms, 13
  - parameters, 14
  - params, 15
  - parameters<- (parameters), 14
  - pars(parameters), 14
  - pars(), 11
  - pars<- (parameters), 14
  - pdims, 15
  - pdims(), 12, 19
  - repair_terms, 16
  - repair_terms(), 3, 5, 21
  - set_parameters(parameters), 14
  - set_pars(parameters), 14
  - subset.term, 17
  - tdim, 18
  - term, 19
  - term(), 6, 16
  - term-object, 20
  - term_vector(term-object), 20
  - term_vector(term-object), 20
  - tindex(tdim), 18
  - unique.term, 20
  - valid_term, 21
  - valid_term(), 9, 16
  - vld_term, 21

23