Package 'ricci'

September 4, 2025

Title Ricci Calculus

Version 0.1.1

Description Provides a compact 'R' interface for performing tensor calculations.
This is achieved by allowing (upper and lower) index labeling of arrays
and making use of Ricci calculus conventions to implicitly trigger
contractions and diagonal subsetting.
Explicit tensor operations, such as addition, subtraction and
multiplication of tensors via the standard operators, raising and lowering
indices, taking symmetric or antisymmetric tensor parts, as well as the
Kronecker product are available. Common tensors like the Kronecker delta,
Levi Civita epsilon, certain metric tensors, the Christoffel symbols,
the Riemann as well as Ricci tensors are provided.
The covariant derivative of tensor fields with respect to any metric
tensor can be evaluated. An effort was made to provide the user with
useful error messages.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, Ryacas, testthat (>= 3.0.0), waldo

Config/testthat/edition 3

Imports calculus, cli, rlang

Depends R (>= 4.1.0)

URL `https://github.com/lschneiderbauer/ricci`,
`https://lschneiderbauer.github.io/ricci/`

BugReports `https://github.com/lschneiderbauer/ricci/issues`

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Author Lukas Schneiderbauer [aut, cre, cph] (ORCID:
<`https://orcid.org/0000-0002-0975-6803`>)

Maintainer Lukas Schneiderbauer <`lukas.schneiderbauer@gmail.com`>

Repository CRAN

Date/Publication 2025-09-04 14:20:02 UTC

Contents

. *Index slot label specification*

Description

This function creates a index slot label specification. Any R symbol can serve as a label. `.()` is typically used in conjunction with %_%.

Usage

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

Arguments

`...`	Index labels separated by commas optionally prefixed by "+" and "-" to indicate the index position (upper and lower respectively). If no prefix is provided, a lower index ("-") is assumed. This argument uses non-standard evaluation: any R symbol that is not a reserved keyword can be used.

Value

A named list of two character vectors representing the index label names and index position.

Examples

```
# three lower index slots
.(i, j, k)

# one lower and upper index
.(i, +j)
```

`as.array.tensor` *Strip array labels*

Description

Converts a `tensor()` to an `array()` by stripping the index labels. An index label order needs to be provided so that the array's `dim()` order is well defined.

Usage

```
## S3 method for class 'tensor'
as.array(
  x,
  index_order = NULL,
  ...,
  arg = "index_order",
  call = rlang::caller_env()
)
```

Arguments

`x`	A labeled array ("tensor" object) created by `%_%` or `tensor()`.
`index_order`	An index specification created with `.()`. The specification needs to match all the labels occurring in `x`. The label order determines the dimension order of the resulting array.
`...`	Not used.
`arg, call`	Used for error handling. Can be ignored by the user.

Value

The tensor components as usual `array()` object without any index labels attached.

Examples

```
array(1:8, dim = c(2, 2, 2)) %_% .(i, +i, k) |> as.array(.(k))
```

asym *Antisymmetric tensor part*

Description

Takes the antisymmetric tensor part of a tensor `x` with respect to the specified indices `...`. An error is thrown if the specified indices do not exist.

Usage

```
asym(x, ...)
```

Arguments

`x`	A labeled tensor object, created by `%_%` or `tensor()`.
`...`	Any number of index expressions. The indices need to occur in `x`.

Value

A modified tensor object.

See Also

Wikipedia: Ricci calculus - Symmetric and antisymmetric parts

Other tensor operations: `Ops.tensor()`, `kron()`, `l()`, `r()`, `subst()`, `sym()`

Examples

```
a <- array(1:4, dim = c(2, 2))
a %_% .(i, j) |> asym(i, j)
```

as_a *Strip array labels*

Description

Converts a `tensor()` to an `array()` by stripping the index labels. An index label order needs to be provided so that the array's `dim()` order is well defined.

Usage

```
as_a(x, ...)
```

Arguments

`x`	A labeled array ("tensor" object) created by `%_%` or `tensor()`.
`...`	Index labels separated by commas optionally prefixed by "+" and "-" to indicate the index position (upper and lower respectively). If no prefix is provided, a lower index ("-") is assumed. This argument uses non-standard evaluation: any R symbol that is not a reserved keyword can be used. The specification needs to match all the labels occurring in `x`. The label order determines the dimension ordering of the resulting array.

Value

A usual `array()` without attached labels. The dimension order is determined by `...`.

See Also

The same functionality is implemented in `as.array.tensor()` but with standard evaluation.

Examples

```
array(1:8, dim = c(2, 2, 2)) %_% .(i, +i, k) |> as_a(k)
```

`at` *Evaluate a symbolic array*

Description

Evaluates a symbolic array at a particular point in parameter space. Partial evaluation is not allowed, all variables/symbols need to be accounted for. The result is a numeric array.

Usage

```
at(x, vars)

## S3 method for class 'array'
at(x, vars)

## S3 method for class 'tensor'
at(x, vars)
```

Arguments

`x`	A symbolic `array()` or a `tensor()`.
`vars`	A named vector with parameter-value assignments. Each named entry represents a substitution of a symbol with the given value.

Value

A numeric `array()` or `tensor()`.

Examples

```
g_sph(3) |> at(c(ph1 = 0, ph2 = 0))
```

`christoffel` *Christoffel symbols*

Description

Provides the Christoffel symbols of the first kind Γ_{ijk} with respect to the Levi Civita connection for a given metric tensor.

Usage

```
christoffel(g)
```

Arguments

g	A covariant metric tensor, a "metric_field" object. See `metric_field()` to create a new metric tensor, or use predefined metrics, e.g. `g_eucl_cart()`.

Details

The Christoffel symbols are a rank 3 array of numbers.

Value

Returns the Christoffel symbols of the first kind Γ_{ijk} as rank 3 `array()`.

See Also

Wikipedia: Christoffel symbols

Other geometric tensors: `ricci()`, `ricci_sc()`, `riemann()`

Examples

```
christoffel(g_eucl_sph(3))
```

`covd` *Covariant Derivative*

Description

Calculates the (symbolic) covariant derivative $\nabla_\rho a^{\nu_1 \nu_2 \cdots}_{\mu_1 \mu_2 \cdots}$ with respect to the Levi Civita connection of any (symbolic) tensor field. The result is a new tensor of one rank higher than the original tensor rank.

Usage

```
covd(x, i, g, act_on = NULL)
```

Arguments

`x`	A labeled tensor object, created by `%_%` or `tensor()`. `covd()` only handles symbolic derivatives, i.e. the tensor components are required to be `character()`-valued and consist of mathematical expressions in terms of coordinates identical to the coordinates used by g.
`i`	An index slot label specification created with `.()`. The number of indices specify the number of covariant derivatives taken in the same order. Each covariant derivative adds one index each with the specified names. After the covariant derivatives are calculated, implicit contraction rules are applied (in case of reoccurring index label names).
`g`	A covariant metric tensor, a "metric_field" object. See `metric_field()` to create a new metric tensor, or use predefined metrics, e.g. `g_eucl_cart()`.
`act_on`	An optional index slot label specification created with `.()` that specifies on which indices the covariant derivative should act on. This might be useful if not all tensor factors are elements of the tangent space of the underlying manifold. If not provided the covariant derivative acts on all indices. If no indices are selected explicitly (with `.()`), the covariant derivative acts like it would on a scalar.

Details

Note that symbolic derivatives are not always completely trustworthy. They usually ignore subtle issues like undefined expressions at certain points. The example $\nabla_a \nabla^a r^{-1}$ from below is telling: The symbolic derivative evaluates to zero identically, although strictly speaking the derivative is not defined at $r = 0$.

Value

The covariant derivative: a new labeled array with one or more additional indices (depending on `i`).

See Also

Wikipedia: Covariant Derivative

Examples

```
options(ricci.auto_simplify = TRUE)

# gradient of "sin(sqrt(x1^2+x2^2+x3^2))" in 3-dimensional euclidean space
covd("sin(x1)", .(k), g = g_eucl_cart(3))

# laplace operator
covd("sin(x1)", .(-k, +k), g = g_eucl_cart(3))
covd("1/r", .(-k, +k), g = g_eucl_sph(3))
```

`d` *Kronecker delta*

Description

Provides a labeled generalized Kronecker delta. In the special case of two labels this represents simply the identity matrix. The Kronecker delta always has an even number of indices. Note that the first half of the tensor labels need to be lowered, while the second half needs upper indices. Otherwise an error is thrown.

Usage

```
d(n)
```

Arguments

`n`	The dimension.

Value

A function that expects index labels (see `.()`) and returns a labeled tensor. The underlying data will differs depending on the number of labels provided.

See Also

Underlying implementation: `calculus::delta()`

Wikipedia: Generalized Kronecker delta

Other tensor symbols: `e()`

Examples

```
d(3)(i, +j)

d(3)(i, j, +k, +l)
```

`e` *Levi-Civita epsilon*

Description

Provides a labeled Levi-Civita epsilon (pseudo) tensor. The indices are required to be lowered. Otherwise an error is thrown.

Usage

```
e(...)
```

Arguments

`...`	Index labels separated by commas optionally prefixed by "+" and "-" to indicate the index position (upper and lower respectively). If no prefix is provided, a lower index ("-") is assumed. This argument uses non-standard evaluation: any R symbol that is not a reserved keyword can be used.

Value

A labeled tensor object. The underlying data will differs depending on the number of labels provided.

See Also

Underlying implementation: `calculus::epsilon()`

Wikipedia: Levi-Civita symbol

Other tensor symbols: `d()`

Examples

```
e(i, j)

e(i, j, k)
```

`expect_tensor_equal` *Does code return the expected value?*

Description

Adds an expectation function that can be used with the testthat package. Compares two tensors and determines whether they are equal or not.

Usage

```
expect_tensor_equal(object, expected, ...)
```

Arguments

`object`, `expected`	Computation and value to compare it to. Both arguments supports limited unquoting to make it easier to generate readable failures within a function or for loop. See quasi_label for more details.
`...`	Arguments passed on to `waldo::compare` `tolerance` If non-`NULL`, used as threshold for ignoring small floating point difference when comparing numeric vectors. Using any non-`NULL` value will cause integer and double vectors to be compared based on their values, not their types, and will ignore the difference between `NaN` and `NA_real_`. It uses the same algorithm as `all.equal()`, i.e., first we generate `x_diff` and `y_diff` by subsetting `x` and `y` to look only locations with differences. Then we check that `mean(abs(x_diff - y_diff)) / mean(abs(y_diff))` (or just `mean(abs(x_diff - y_diff))` if `y_diff` is small) is less than `tolerance`. `max_diffs` Control the maximum number of differences shown. The default shows 10 differences when run interactively and all differences when run in CI. Set `max_diffs = Inf` to see all differences. `ignore_srcref` Ignore differences in function `srcref`s? `TRUE` by default since the `srcref` does not change the behaviour of a function, only its printed representation. `ignore_encoding` Ignore string encoding? `TRUE` by default, because this is R's default behaviour. Use `FALSE` when specifically concerned with the encoding, not just the value of the string.

Value

The actual value invisibly.

`g_eucl_cart` *Euclidean metric tensor*

Description

Provides the Euclidean metric tensor of $\mathbb{E}^n$. `g_eucl_cart()` returns a numeric (constant) tensor in Cartesian coordinates,

$$ds^2 = \sum_{i=1}^{n} dx_i^2$$

while `g_eucl_sph()` returns a symbolic tensor field in generalized spherical coordinates $r, \phi_1, \phi_2, ..., \phi_{n-1}$.

$$ds^2 = dr^2 + r^2 d\Omega^2$$

Usage

```
g_eucl_cart(n, coords = paste0("x", 1:n))

g_eucl_sph(n, coords = c("r", paste0("ph", 1:(n - 1))))
```

Arguments

n	The dimension of the metric tensor.
coords	A character vector of coordinate names. The length needs to match the tensor dimensions.

Details

As usual, spherical coordinates are degenerate at $r = 0$ and $\phi_l = 0$, so be careful around those points.

Value

The covariant metric tensor as array imputed with coordinate names.

See Also

Wikipedia: Euclidean metric tensor

Other metric tensors: `g_mink_cart()`, `g_sph()`, `g_ss()`, `metric_field()`

Examples

```
g_eucl_cart(3)
g_eucl_cart(3) %_% .(+i, +j)
g_eucl_sph(3)
g_eucl_sph(3) %_% .(+i, +j)
```

`g_mink_cart` *Minkowski metric tensor*

Description

`g_mink_cart()` provides the covariant metric tensor in `n` dimensions in Cartesian coordinates with signature $(-1, 1, 1, ...)$.

$$ds^2 = -dx_0^2 + \sum_{i=1}^{n-1} dx_i^2$$

`g_mink_sph()` provides the same tensor where the spatial part uses spherical coordinates.

$$ds^2 = -dt^2 + dr^2 + r^2 d\Omega^2$$

Usage

```
g_mink_cart(n, coords = paste0("x", 1:n - 1))

g_mink_sph(n, coords = c("t", "r", paste0("ph", 1:(n - 2))))
```

Arguments

`n`	The dimension of the metric tensor.
`coords`	A character vector of coordinate names. The length needs to match the tensor dimensions.

Value

The covariant metric tensor as array imputed with coordinate names.

See Also

Wikipedia Minkowski metric tensor

Other metric tensors: `g_eucl_cart()`, `g_sph()`, `g_ss()`, `metric_field()`

Examples

```
g_mink_cart(4)
g_mink_cart(4) %_% .(+i, +j)
g_mink_sph(4)
g_mink_sph(4) %_% .(+i, +j)
```

`g_sph` *Metric tensor of the sphere*

Description

Provides the metric tensor of the sphere S^n with radius 1. `g_sph()` returns a symbolic tensor field in generalized spherical coordinates $\phi_1, \phi_2, ..., \phi_{n-1}$.

$$d\Omega^2 = d\phi_1^2 + \sum_{i=1}^{n-1} \prod_{m=1}^{i-1} sin(\phi_m)^2 d\phi_i^2$$

Usage

```
g_sph(n, coords = paste0("ph", 1:n))
```

Arguments

`n`	The dimension of the metric tensor.
`coords`	A character vector of coordinate names. The length needs to match the tensor dimensions.

Details

As usual, spherical coordinates are degenerate at $\phi_l = 0$, so be careful around those points.

Value

The covariant metric tensor as array imputed with coordinate names.

See Also

Wikipedia: Sphere

Other metric tensors: `g_eucl_cart()`, `g_mink_cart()`, `g_ss()`, `metric_field()`

Examples

```
g_sph(3)
g_sph(3) %_% .(+i, +j)
```

`g_ss` *Schwarzschild metric tensor*

Description

Provides the metric tensor of the Einstein equation's Schwarzschild solution in Schwarzschild coordinates where the Schwarzschild radius r_s is set to 1.

$$ds^2 = -\left(1 - \frac{r_s}{r}\right) dt^2 + \left(1 - \frac{r_s}{r}\right)^{-1} dr^r + r^2 d\Omega^2$$

Usage

```
g_ss(n, coords = c("t", "r", paste0("ph", 1:(n - 2))))
```

Arguments

`n`	The dimension of the metric tensor.
`coords`	A character vector of coordinate names. The length needs to match the tensor dimensions.

Details

Note that Schwarzschild coordinates become singular at the Schwarzschild radius (event horizon) $r = r_s = 1$ and at $r = 0$.

Value

The covariant metric tensor as array imputed with coordinate names.

See Also

Wikipedia: Schwarzschild metric

Other metric tensors: `g_eucl_cart()`, `g_mink_cart()`, `g_sph()`, `metric_field()`

Examples

```
g_ss(4)
g_ss(4) %_% .(+i, +j)
```

`kron` *Kronecker product*

Description

A Kronecker product is simply a tensor product whose underlying vector space basis is relabeled. In the present context this is realized by combining multiple index labels into one. The associated dimension to the new label is then simply the product of the dimensions associated to the old index labels respectively.

Usage

```
kron(x, ...)
```

Arguments

`x`	A labeled tensor object, created by `%_%` or `tensor()`.
`...`	Any number of expressions (separated by a comma) of the form `.(<label1>, <label2>, ..., <labeln`

Value

A modified tensor object.

See Also

Wikipedia: Kronecker Product

Other tensor operations: `Ops.tensor()`, `asym()`, `l()`, `r()`, `subst()`, `sym()`

Examples

```
a <- array(1:8, dim = c(2, 2, 2))
a %_% .(i, j, k) |> kron(.(i, j) -> l)
```

l *Lower tensor indices*

Description

`l()` lowers specified tensor indices using a covariant metric tensor provided in `g`. Note that the order of indices is not preserved due to performance reasons. An error is thrown if the specified indices do not exist or are not in the correct position.

Usage

```
l(x, ..., g = NULL)
```

Arguments

`x`	A labeled tensor object, created by `%_%` or `tensor()`.
`...`	Any number of index expressions. The indices need to occur in `x`.
`g`	A covariant metric tensor, a "metric_field" object. See `metric_field()` to create a new metric tensor, or use predefined metrics, e.g. `g_eucl_cart()`. If no metric tensor is provided, indices are raised/lowered with the identity matrix.

Value

A modified tensor object.

See Also

Other tensor operations: `Ops.tensor()`, `asym()`, `kron()`, `r()`, `subst()`, `sym()`

metric_field *Create a metric tensor field*

Description

Metric tensors are an essential ingredient of (Pseudo-) Riemannian manifolds and define distance relations between points. They are used to define geometric tensors such as e.g. the Ricci curvature `ricci()`, and a metric connection, i.e. a covariant derivative. They are also essential for raising and lowering indices of tensor fields correctly when using non-flat coordinates.

Usage

```
metric_field(metric, metric_inv, coords)
```

Arguments

`metric`	A `nxn` matrix / array representing the covariant metric tensor components. The components are usually expressions as character strings formed from coordinates, since numeric values can only represent constant tensor fields.
`metric_inv`	A `nxn` matrix / array representing the contraviant metric tensor components, i.e. the inverse matrix of the covariant metric tensor component matrix.
`coords`	A character vector of `n` coordinate names that are used in the component expressions. This information is essential for forming symbolic derivatives.

Value

An object of class `c("metric_field", "array")` that represents the components of a metric tensor on a (Pseudo-) Riemannian manifold in a certain coordinate system specified by `coords`.

See Also

Wikipedia: Metric tensor

Other metric tensors: `g_eucl_cart()`, `g_mink_cart()`, `g_sph()`, `g_ss()`

`Ops.tensor` *Arithmetic tensor operations*

Description

Once a labeled array (tensor) has been defined, tensor arithmetic operations can be carried out with the usual `+`, `-`, `*`, `/`, and `==` symbols.

Usage

```
## S3 method for class 'tensor'
Ops(e1, e2)
```

Arguments

`e1, e2`	Labeled arrays created with `%_%`.

Value

A resulting labeled array in case of `+`, `-`, `*`, `/`. `TRUE` or `FALSE` in case of `==`.

Addition and Subtraction

Addition and subtraction requires the two tensors to have an equal index structure, i.e. the index names their position and the dimensions associated to the index names have to agree. The index order does not matter, the operation will match dimensions by index name.

Multiplication

Tensor multiplication takes into account implicit Ricci calculus rules depending on index placement.

- Equal-named and opposite-positioned dimensions are contracted.
- Equal-named and equal-positioned dimensions are subsetted.
- The result is an outer product for distinct index names.

Division

Division performs element-wise division. If the second argument is a scalar, each element is simply divided by the scalar. Similar to addition and subtraction, division requires the two tensors to have an equal index structure, i.e. the index names their position and the dimensions associated to the index names have to agree.

Equality check

A tensor $a_{i_1 i_2 \ldots}$ is equal to a tensor $b_{j_1 j_2 \ldots}$ if and only if the index structure agrees and all components are equal.

See Also

Other tensor operations: `asym()`, `kron()`, `l()`, `r()`, `subst()`, `sym()`

Examples

```
a <- array(1:4, c(2, 2))
b <- array(3 + 1:4, c(2, 2))

# addition
a %_% .(i, j) + b %_% .(j, i)

# multiplication
a %_% .(i, j) * b %_% .(+i, k)

# division
a %_% .(i, j) / 10

# equality check
a %_% .(i, j) == a %_% .(i, j)
a %_% .(i, j) == a %_% .(j, i)
a %_% .(i, j) == b %_% .(i, j)

# this will err because index structure does not agree
try(a %_% .(i, j) == a %_% .(k, j))
```

r *Raise tensor indices*

Description

`r()` raises specified tensor indices using a covariant metric tensor provided in g. Note that the order of indices is not preserved due to performance reasons. An error is thrown if the specified indices do not exist or are not in the correct position.

Usage

```
r(x, ..., g = NULL)
```

Arguments

`x`	A labeled tensor object, created by `%_%` or `tensor()`.
`...`	Any number of index expressions. The indices need to occur in `x`.
`g`	A covariant metric tensor, a "metric_field" object. See `metric_field()` to create a new metric tensor, or use predefined metrics, e.g. `g_eucl_cart()`. If no metric tensor is provided, indices are raised/lowered with the identity matrix.

Value

A modified tensor object.

See Also

Other tensor operations: `Ops.tensor()`, `asym()`, `kron()`, `l()`, `subst()`, `sym()`

ricci *Ricci curvature tensor*

Description

Provides the covariant Ricci curvature tensor $R_{ij} = R^s_{\ isj}$.

Usage

```
ricci(g)
```

Arguments

`g`	A covariant metric tensor, a "metric_field" object. See `metric_field()` to create a new metric tensor, or use predefined metrics, e.g. `g_eucl_cart()`.

Value

Returns the covariant Ricci curvature tensor R_{ij} as rank 2 `array()`.

See Also

Wikipedia: Ricci curvature tensor

Other geometric tensors: `christoffel()`, `ricci_sc()`, `riemann()`

Examples

```
ricci(g_eucl_sph(3))
```

`ricci_sc` *Ricci scalar*

Description

Provides the Ricci scalar R.

Usage

```
ricci_sc(g)
```

Arguments

g	A covariant metric tensor, a "metric_field" object. See `metric_field()` to create a new metric tensor, or use predefined metrics, e.g. `g_eucl_cart()`.

Value

Returns the Ricci scalar R as single number/expression.

See Also

Wikipedia: Ricci scalar

Other geometric tensors: `christoffel()`, `ricci()`, `riemann()`

Examples

```
ricci_sc(g_eucl_sph(3))
```

riemann — *Riemann curvature tensor*

Description

Provides the covariant Riemann curvature tensor R_{ijkl}.

Usage

```
riemann(g)
```

Arguments

`g`	A covariant metric tensor, a "metric_field" object. See `metric_field()` to create a new metric tensor, or use predefined metrics, e.g. `g_eucl_cart()`.

Value

Returns the covariant Riemann curvature tensor R_{ijkl} as rank 4 `array()`.

See Also

Wikipedia: Riemann curvature tensor

Other geometric tensors: `christoffel()`, `ricci()`, `ricci_sc()`

Examples

```
riemann(g_eucl_sph(3))
```

simplify — *Simplify symbolic expressions*

Description

Attempts to simplify expressions in an array or tensor. Non-array objects are coerced to arrays with `as.array()`.

Usage

```
simplify(x)
```

Arguments

`x`	A character `array()` or `tensor()` consisting of mathematical expressions.

Details

Instead of using an explicit call to `simplify()` you also have the option to enable automatic simplification via `option(ricci.auto_simplify = TRUE)`. Note however that this comes at a significant performance cost.

This operation requires the Ryacas package.

Value

A character `array()` or `tensor()` of the same form, potentially with simplified expressions.

Examples

```
simplify("x + y - x")
```

`subst` *Substitute tensor labels*

Description

Substitutes tensor labels with other labels. This is simply a renaming procedure. The operation might trigger implicit diagonal subsetting. An error is thrown if the specified indices do not exist.

Usage

```
subst(x, ...)
```

Arguments

`x`	A labeled tensor object, created by `%_%` or `tensor()`.
`...`	Any number of expressions (separated by a comma) of the form `<label1> -> <label2>`.

Value

A modified tensor object.

See Also

Other tensor operations: `Ops.tensor()`, `asym()`, `kron()`, `l()`, `r()`, `sym()`

sym *Symmetric tensor part*

Description

Takes the symmetric tensor part of a tensor `x` with respect to the specified indices `....` An error is thrown if the specified indices do not exist.

Usage

```
sym(x, ...)
```

Arguments

`x`	A labeled tensor object, created by `%_%` or `tensor()`.
`...`	Any number of index expressions. The indices need to occur in `x`.

Value

A modified tensor object.

See Also

Wikipedia: Ricci calculus - Symmetric and antisymmetric parts

Other tensor operations: `Ops.tensor()`, `asym()`, `kron()`, `l()`, `r()`, `subst()`

Examples

```
a <- array(1:4, dim = c(2, 2))
a %_% .(i, j) |> sym(i, j)
```

tensor *Create a labeled array (tensor)*

Description

Creates a labeled array (tensor) from an array. `%_%` and `tensor()` serve the same purpose, but typically usage of `%_%` is preferred due to brevity. `tensor()` is exported to provide a standard-evaluation interface as well which might be useful under some circumstances.

Usage

```
tensor(a, index_names, index_positions, call = NULL)

a %_% i
```

Arguments

`a`	An array or any object that can be coerced to an array via `as.array()`.
`index_names`	A character vector of index names / labels.
`index_positions`	A character vector of index positions with two allowed values "+" and "-", for "upper" and "lower" position respectively. The length of `index_positions` needs to agree with the length of `index_names`.
`call`	For internal use only.
`i`	An index slot label specification created with `.()`.

Value

A labeled tensor object of class `"tensor"`, an `array()` with attached dimension labels. Note that the index structure of the resulting tensor does not necessarily have to match `i`. In case implicit calculations are already triggered (e.g. contractions) the index structure reflects the resulting tensor.

Examples

```
a <- array(1:4, dim = c(2, 2))
a %_% .(i, j)
```

Index