Package 'cgraph'

February 9, 2020

Type Package

Title Computational Graphs

Version 6.0.1

Author Ron Triepels

Maintainer Ron Triepels <dev@cgraph.org>

URL https://cgraph.org/

BugReports https://github.com/triepels/cgraph/issues

Description Allows to create, evaluate, and differentiate computational graphs in R. A computational graph is a graph representation of a multivariate function decomposed by its (elementary) operations. Nodes in the graph represent arrays while edges represent dependencies among the arrays. An advantage of expressing a function as a computational graph is that this enables to differentiate the function by automatic differentiation. The 'cgraph' package supports various operations including basic arithmetic, trigonometry operations, and linear algebra operations. It differentiates computational graphs by reverse automatic differentiation. The flexible architecture of the package makes it applicable to solve a variety of problems including local sensitivity analysis, gradient-based optimization, and machine learning.

License Apache License 2.0

Encoding UTF-8

LazyData true

Suggests testthat

RoxygenNote 7.0.2

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-02-09 16:50:05 UTC

R topics documented:

`cg_abs` *Absolute Value*

Description

Calculate `abs(x)`.

Usage

```
cg_abs(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

abs

cg_acos *Inverse Cosine*

Description

Calculate `acos(x)`.

Usage

```
cg_acos(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

acos

cg_acosh *Inverse Hyperbolic Cosine*

Description

Calculate `acosh(x)`.

Usage

```
cg_acosh(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

acosh

`cg_add` *Add*

Description

Calculate `x + y`.

Usage

```
cg_add(x, y, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`y`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

add

cg_asin *Inverse Sine*

Description

Calculate `asin(x)`.

Usage

```
cg_asin(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

asin

cg_asinh *Inverse Hyperbolic Sine*

Description

Calculate `asinh(x)`.

Usage

```
cg_asinh(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

asinh

`cg_as_double` *Coerce to a Numerical Vector*

Description

Coerce `x` to a one-dimensional numerical vector.

Usage

```
cg_as_double(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical matrix or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This function is identical to `cg_as_numeric`.

Author(s)

Ron Triepels

See Also

as.double

cg_as_numeric *Coerce to a Numerical Vector*

Description

Coerce `x` to a one-dimensional numerical vector.

Usage

```
cg_as_numeric(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical matrix or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This function is identical to `cg_as_double`.

Author(s)

Ron Triepels

See Also

as.numeric

cg_atan *Inverse Tangent*

Description

Calculate `atan(x)`.

Usage

```
cg_atan(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

atan

`cg_atanh` *Inverse Hyperbolic Tangent*

Description

Calculate `atanh(x)`.

Usage

```
cg_atanh(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

atanh

cg_colmeans *Column Means*

Description

Calculate `colMeans(x)`.

Usage

```
cg_colmeans(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical matrix or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function colMeans is called without changing the default value of argument `na.rm` and `dims`.

Author(s)

Ron Triepels

See Also

colMeans

cg_colsums *Column Sums*

Description

Calculate `colSums(x)`.

Usage

```
cg_colsums(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical matrix or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function colSums is called without changing the default value of argument `na.rm` and `dims`.

Author(s)

Ron Triepels

See Also

colSums

`cg_constant` *Add Constant*

Description

Add a constant node to the active graph.

Usage

```
cg_constant(value, name = NULL)
```

Arguments

`value`	R object, value of the node.
`name`	character scalar, name of the node (optional). In case argument `name` is missing, the node is added to the graph under an automatically generated name.

Value

cg_node object.

Note

Constant nodes are ignored when differentiating a graph.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()

# Add a constant with value 1 and name 'a' to the graph.
a <- cg_constant(1, name = "a")
```

---

cg_cos *Cosine*

Description

Calculate `cos(x)`.

Usage

```
cg_cos(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

cos

cg_cosh *Hyperbolic Cosine*

Description

Calculate `cosh(x)`.

Usage

```
cg_cosh(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

cosh

cg_crossprod *Matrix Crossproduct*

Description

Calculate `crossprod(x,y)`.

Usage

```
cg_crossprod(x, y = x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical matrix.
y	either a cg_node object or a numerical matrix (optional).
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

crossprod

`cg_dim` *Dimensions of an Array*

Description

Calculate `dim(x)`.

Usage

```
cg_dim(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This operator is not differentiable. Any attempt to differentiate this operator will result in an error.

Author(s)

Ron Triepels

See Also

dim

cg_div *Divide*

Description

Calculate `x / y`.

Usage

```
cg_div(x, y, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`y`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

divide

cg_exp *Exponential Function*

Description

Calculate `exp(x)`.

Usage

```
cg_exp(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

exp

`cg_function` *Create function*

Description

Initialize a new function that can be used by operators in a graph.

Usage

```
cg_function(def, grads = list())
```

Arguments

`def`	function, the definition of the function.
`grads`	list of functions, the gradient functions with respect to each input (optional).

Value

cg_function object.

Note

If the function consumes any inputs, then the gradient function with respect to these inputs must be provided to argument `grads`. These gradients must be a function of each input's gradient and take as arguments the inputs of the function including argument `value` and `grad`. These latter two arguments evaluate to the value of the function and its gradient respectively at run-time.

Author(s)

Ron Triepels

Examples

```
#' # Create a custom negation function
f <- cg_function(
    def = function(x) -x,
    grads = list(function(x, value, grad) -grad)
)
```

cg_graph *Computational Graph*

Description

Initialize a computational graph.

Usage

```
cg_graph(eager = TRUE)
```

Arguments

eager	logical scalar, should new nodes added to the graph be evaluated eagerly? Defaults to TRUE.

Value

cg_graph object.

Note

The graph is automatically set to be the active graph.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()
```

cg_graph_backward *Backward Pass*

Description

Perform a backward pass to evaluate the partial derivatives of a given target node with respect to the nodes in a graph.

Usage

```
cg_graph_backward(graph, target, index = NULL)
```

Arguments

graph	cg_graph object, graph that is differentiated.
target	cg_node object, node in the graph that is differentiated. Alternatively, argument target can be a character scalar denoting the name of the node in the graph that is differentiated.
index	numerical scalar, index of the target node that is differentiated. Defaults to NULL (i.e. all elements are differentiated element-wise).

Value

None.

Note

All nodes required to compute the target node must first have been evaluated by calling cg_graph_forward. The target node is only differenated with respect to those nodes on which it directly or indirectly depends.

In case the value of the target node is a vector or an array, argument index can be used to specify which element of the vector or array is differentiated.

The derivatives have the same shape as the values of the nodes. They can be retrieved via the grad data member of a cg_node object.

If the name of the target node is supplied to argument target, a linear search is performed to retrieve the node from the graph. In case multiple nodes share the same name, the last node added to the graph is retrieved. Please note that this linear search can become relatively expensive for large graphs.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()

# Add an input
a <- cg_input(name = "a")

# Add a parameter
b <- cg_parameter(4, name = "b")

# Perform some operations
c <- cg_sin(a) + cg_cos(b) - cg_tan(a)

# Set a equal to 2
a$value <- 2

# Perform forward pass
cg_graph_forward(graph, c)
```

```
# Perform backward pass
cg_graph_backward(graph, c)

# Retrieve the derivative of c with respect to b
b$grad
```

`cg_graph_forward` *Forward Pass*

Description

Perform a forward pass to evaluate a given target node in a graph.

Usage

```
cg_graph_forward(graph, target)
```

Arguments

`graph`	cg_graph object, graph that is evaluated.
`target`	cg_node object, node in the graph that is evaluated. Alternatively, argument `target` can be a character scalar denoting the name of the node in the graph that is evaluated.

Value

None.

Note

All nodes required to compute the target node must have a value or their value must be able to be computed at run-time. Only those nodes needed to compute the target node (including the target itself) are evaluated.

The value of a node can be retrieved via the `values` data member of a `cg_node` object.

If the name of the target node is supplied to argument `target`, a linear search is performed to retrieve the node from the graph. In case multiple nodes share the same name, the last node added to the graph is retrieved. Please note that this linear search can become relatively expensive for large graphs.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()

# Add an input
a <- cg_input(name = "a")

# Square the input (i.e. b = a^2)
b <- cg_pow(a, 2, name = "b")

# Set a equal to 2
a$value <- 2

# Perform forward pass
cg_graph_forward(graph, b)

# Retrieve the value of b
b$value
```

---

cg_graph_get *Retrieve Node*

Description

Retrieve a node from a graph by name.

Usage

```
cg_graph_get(graph, name)
```

Arguments

graph	cg_graph object, graph containing the node to be retrieved.
name	character scalar, name of the node to be retrieved.

Value

cg_node object.

Note

In case multiple nodes share the same name, the last node added to the graph is retrieved.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()

# Add an input
a <- cg_input(name = "a")

# Retrieve input a
b <- cg_graph_get(graph, "a")

# Check equality
identical(a, b)
```

`cg_input` *Add Input*

Description

Add an input node to the active graph.

Usage

```
cg_input(name = NULL)
```

Arguments

`name`	character scalar, name of the node (optional). In case argument `name` is missing, the node is added to the graph under an automatically generated name.

Value

cg_node object.

Note

Inputs cannot be assigned a value upon creation. Instead, they behave as placeholders. You can use data member `value` of a cg_node object to retrieve or change its value.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()

# Add an input with name 'a' to the graph.
a <- cg_input(name = "a")

# Set the value to 2
a$value <- 2
```

`cg_length` *Length of an Object*

Description

Calculate `length(x)`.

Usage

```
cg_length(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This operator is not differentiable. Any attempt to differentiate this operator will result in an error.

Author(s)

Ron Triepels

See Also

length

cg_linear *Linear Transformation*

Description

Calculate `x %*% y + c(z)`.

Usage

```
cg_linear(x, y, z, name = NULL)
```

Arguments

x	either a cg_node object or a numerical matrix.
y	either a cg_node object or a numerical matrix.
z	either a cg_node object or a numerical vector.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This function is equivalent to `cg_matmul(x,y) + cg_as_numeric(z)`.

Author(s)

Ron Triepels

cg_ln *Natural Logarithm*

Description

Calculate `log(x)`.

Usage

```
cg_ln(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

log

`cg_log10` *Logarithm Base 10*

Description

Calculate `log10(x)`.

Usage

```
cg_log10(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

log10

cg_log2 *Logarithm Base 2*

Description

Calculate `log2(x)`.

Usage

```
cg_log2(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

log2

cg_matmul *Matrix Multiplication*

Description

Calculate `x %*% y`.

Usage

```
cg_matmul(x, y, name = NULL)
```

Arguments

x	either a cg_node object or a numerical matrix.
y	either a cg_node object or a numerical matrix.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

matmult

`cg_max` *Maxima*

Description

Calculate `max(x)`.

Usage

```
cg_max(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function max is called without changing the default value of argument `na.rm`.

Author(s)

Ron Triepels

See Also

max

cg_mean *Arithmetic Mean*

Description

Calculate `mean(x)`.

Usage

```
cg_mean(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function mean is called without changing the default value of argument `trim` and `na.rm`.

Author(s)

Ron Triepels

See Also

mean

cg_min *Minima*

Description

Calculate `min(x)`.

Usage

```
cg_min(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function min is called without changing the default value of argument `na.rm`.

Author(s)

Ron Triepels

See Also

min

`cg_mul` *Multiply*

Description

Calculate `x * y`.

Usage

```
cg_mul(x, y, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`y`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

multiply

cg_ncol *Number of Columns of an Array*

Description

Calculate `ncol(x)`.

Usage

```
cg_ncol(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This operator is not differentiable. Any attempt to differentiate this operator will result in an error.

Author(s)

Ron Triepels

See Also

ncol

cg_neg *Negative*

Description

Calculate `-x`.

Usage

```
cg_neg(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

negative

`cg_nrow` *Number of Rows of an Array*

Description

Calculate `nrow(x)`.

Usage

```
cg_nrow(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This operator is not differentiable. Any attempt to differentiate this operator will result in an error.

Author(s)

Ron Triepels

See Also

nrow

cg_operator *Add Operator*

Description

Add an operation node to the active graph.

Usage

```
cg_operator(fun, inputs, name = NULL)
```

Arguments

fun	cg_function object, function evaluated by the node.
inputs	list, the nodes that are consumed by the operation.
name	character scalar, name of the node (optional). In case argument name is missing, the node is added to the graph under an automatically generated name.

Value

cg_node object.

Note

Any objects that are supplied to argument inputs that are not cg_node objects are implicitly coerced to cg_constant objects.

The elements of argument input can be named to control how the arguments of the function provided to argument fun are machted when the function is evaluated. In case no names are provided, arguments are matched positionally.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()

# Create a custom negation function
f <- cg_function(
    def = function(x) -x,
    grads = list(function(x, val, grad) -grad)
)

# Add a an operator with the negation function to the graph.
a <- cg_operator(f, list(10), name = "a")
```

cg_parameter *Add Parameter*

Description

Add a parameter node to the active graph.

Usage

```
cg_parameter(value, name = NULL)
```

Arguments

value	numerical vector or array, value of the node.
name	character scalar, name of the node (optional). In case argument name is missing, the node is added to the graph under an automatically generated name.

Value

cg_node object.

Note

Parameters are assumed to be subject to some optimization process. Hence, their value might change over time. You can use data member value of a cg_node object to retrieve or change its value.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()

# Add a parameter with value 1 and name 'a' to the graph.
a <- cg_parameter(1, name = "a")
```

cg_pmax *Parallel Maxima*

Description

Calculate `pmax(x,y)`.

Usage

```
cg_pmax(x, y, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`y`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function pmax is called without changing the default value of argument `na.rm`.

Author(s)

Ron Triepels

See Also

pmax

cg_pmin *Parallel Minima*

Description

Calculate `pmin(x,y)`.

Usage

```
cg_pmin(x, y, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
y	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function pmin is called without changing the default value of argument na.rm.

Author(s)

Ron Triepels

See Also

pmin

---

cg_pos *Positive*

Description

Calculate x.

Usage

```
cg_pos(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

positive

cg_pow *Power*

Description

Calculate `x ^ y`.

Usage

```
cg_pow(x, y, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
y	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

power

cg_prod *Product of Vector Elements*

Description

Calculate `prod(x)`.

Usage

```
cg_prod(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

In contrast to the base prod function, this function only accepts a single argument.

Function prod is called without changing the default value of argument `na.rm`.

Author(s)

Ron Triepels

See Also

prod

`cg_rowmeans` *Row Means*

Description

Calculate `rowMeans(x)`.

Usage

```
cg_rowmeans(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical matrix or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function rowMeans is called without changing the default value of argument `na.rm` and `dims`.

Author(s)

Ron Triepels

See Also

rowMeans

cg_rowsums *Row Sums*

Description

Calculate `rowSums(x)`.

Usage

```
cg_rowsums(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical matrix or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

Function rowSums is called without changing the default value of argument `na.rm` and `dims`.

Author(s)

Ron Triepels

See Also

rowSums

cg_session_graph *Get Active Graph*

Description

Get the graph that is currently active.

Usage

```
cg_session_graph()
```

Value

cg_graph object.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph <- cg_graph()

# Retrieve the graph from the session
cg_session_graph()
```

---

cg_session_set_graph *Change Active Graph*

Description

Set a graph to be the active graph.

Usage

```
cg_session_set_graph(graph)
```

Arguments

graph	cg_graph object, the graph that is activated.

Value

none.

Note

Any nodes that are created are automatically added to the active graph. This also applies to operations that are created by overloaded S3 functions that do not follow the cg_<name> naming convention (such as primitive inflix functions '+' and '-').

Only one graph can be active at a time. The active graph can be changed by calling this function on another cg_graph object.

Author(s)

Ron Triepels

Examples

```
# Initialize a computational graph
graph1 <- cg_graph()

# Initialize another computational graph. It becomes the active graph.
graph2 <- cg_graph()

# Set graph1 to be the active graph
cg_session_set_graph(graph1)
```

`cg_sigmoid` *Sigmoid*

Description

Calculate `1 / (1 + exp(-x))`.

Usage

```
cg_sigmoid(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

`cg_sin` *Sine*

Description

Calculate `sin(x)`.

Usage

```
cg_sin(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

sin

---

cg_sinh *Hyperbolic Sine*

Description

Calculate `sinh(x)`.

Usage

```
cg_sinh(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

sinh

cg_sqrt *Square Root*

Description

Calculate `sqrt(x)`.

Usage

```
cg_sqrt(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

sqrt

cg_square *Square*

Description

Calculate `x ^ 2`.

Usage

```
cg_square(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This function is equivalent to `cg_pow(x,2)`.

Author(s)

Ron Triepels

See Also

square

`cg_sub` *Subtract*

Description

Calculate `x -y`.

Usage

```
cg_sub(x, y, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`y`	either a cg_node object or a numerical vector or array.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

subtract

cg_subset1 *Subset*

Description

Calculate `x[...]`.

Usage

```
cg_subset1(x, ..., name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical vector or array.
`...`	either cg_node objects or numerical scalars that are passed on to the `` `[` `` function.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This operator is not differentiable with respect to the arguments provided to `....` Any attempt to differentiate this operator with respect to these arguments results in an error.

Author(s)

Ron Triepels

See Also

subset

cg_subset2 *Subset*

Description

Calculate `x[[...]]`.

Usage

```
cg_subset2(x, ..., name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
...	either cg_node objects or numerical scalars that are passed on to the `[[` function.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

This operator is not differentiable with respect to the arguments provided to .... Any attempt to differentiate this operator with respect to these arguments results in an error.

Author(s)

Ron Triepels

See Also

subset

---

cg_sum *Sum of Vector Elements*

Description

Calculate sum(x).

Usage

```
cg_sum(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Note

In contrast to the base sum function, this function only accepts a single argument.

Function sum is called without changing the default value of argument na.rm.

Author(s)

Ron Triepels

See Also

sum

`cg_t` *Matrix Transpose*

Description

Calculate `t(x)`.

Usage

```
cg_t(x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical matrix.
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

t

cg_tan *Tangent*

Description

Calculate `tan(x)`.

Usage

```
cg_tan(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

tan

cg_tanh *Hyperbolic Tangent*

Description

Calculate `tanh(x)`.

Usage

```
cg_tanh(x, name = NULL)
```

Arguments

x	either a cg_node object or a numerical vector or array.
name	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

tanh

`cg_tcrossprod` *Transpose Matrix Crossproduct*

Description

Calculate `tcrossprod(x,y)`.

Usage

```
cg_tcrossprod(x, y = x, name = NULL)
```

Arguments

`x`	either a cg_node object or a numerical matrix.
`y`	either a cg_node object or a numerical matrix (optional).
`name`	character scalar, name of the operation (optional).

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

tcrossprod

Index