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

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

R topics documented:

  cg_abs ................................................................. 3
gc_acos ................................................................. 4
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cg_acosh</td>
<td>4</td>
</tr>
<tr>
<td>cg_add</td>
<td>5</td>
</tr>
<tr>
<td>cg_asin</td>
<td>6</td>
</tr>
<tr>
<td>cg_asinh</td>
<td>6</td>
</tr>
<tr>
<td>cg_as_double</td>
<td>7</td>
</tr>
<tr>
<td>cg_as_numeric</td>
<td>8</td>
</tr>
<tr>
<td>cg_atan</td>
<td>8</td>
</tr>
<tr>
<td>cg_atanh</td>
<td>9</td>
</tr>
<tr>
<td>cg_colmeans</td>
<td>10</td>
</tr>
<tr>
<td>cg_colsums</td>
<td>10</td>
</tr>
<tr>
<td>cg_constant</td>
<td>11</td>
</tr>
<tr>
<td>cg_cos</td>
<td>12</td>
</tr>
<tr>
<td>cg_cosh</td>
<td>13</td>
</tr>
<tr>
<td>cg_crossprod</td>
<td>13</td>
</tr>
<tr>
<td>cg_dim</td>
<td>14</td>
</tr>
<tr>
<td>cg_div</td>
<td>15</td>
</tr>
<tr>
<td>cg_exp</td>
<td>15</td>
</tr>
<tr>
<td>cg_function</td>
<td>16</td>
</tr>
<tr>
<td>cg_graph</td>
<td>17</td>
</tr>
<tr>
<td>cg_graph_backward</td>
<td>17</td>
</tr>
<tr>
<td>cg_graph_forward</td>
<td>19</td>
</tr>
<tr>
<td>cg_graph_get</td>
<td>20</td>
</tr>
<tr>
<td>cg_input</td>
<td>21</td>
</tr>
<tr>
<td>cg_length</td>
<td>22</td>
</tr>
<tr>
<td>cg_linear</td>
<td>23</td>
</tr>
<tr>
<td>cg_ln</td>
<td>23</td>
</tr>
<tr>
<td>cg_log10</td>
<td>24</td>
</tr>
<tr>
<td>cg_log2</td>
<td>25</td>
</tr>
<tr>
<td>cg_matmul</td>
<td>25</td>
</tr>
<tr>
<td>cg_max</td>
<td>26</td>
</tr>
<tr>
<td>cg_mean</td>
<td>27</td>
</tr>
<tr>
<td>cg_min</td>
<td>27</td>
</tr>
<tr>
<td>cg_mul</td>
<td>28</td>
</tr>
<tr>
<td>cg_ncol</td>
<td>29</td>
</tr>
<tr>
<td>cg_neg</td>
<td>29</td>
</tr>
<tr>
<td>cg_nrow</td>
<td>30</td>
</tr>
<tr>
<td>cg_operator</td>
<td>31</td>
</tr>
<tr>
<td>cg_parameter</td>
<td>32</td>
</tr>
<tr>
<td>cg_pmax</td>
<td>33</td>
</tr>
<tr>
<td>cg_pmin</td>
<td>33</td>
</tr>
<tr>
<td>cg_pos</td>
<td>34</td>
</tr>
<tr>
<td>cg_pow</td>
<td>35</td>
</tr>
<tr>
<td>cg_prod</td>
<td>35</td>
</tr>
<tr>
<td>cg_rowmeans</td>
<td>36</td>
</tr>
<tr>
<td>cg_rowsums</td>
<td>37</td>
</tr>
<tr>
<td>cg_session_graph</td>
<td>37</td>
</tr>
<tr>
<td>cg_session_set_graph</td>
<td>38</td>
</tr>
<tr>
<td>cg_sigmoid</td>
<td>39</td>
</tr>
</tbody>
</table>
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).
\textit{cg_add}

\textbf{Value}

cg\_operator object.

\textbf{Author(s)}

Ron Triepels

\textbf{See Also}

\texttt{acosh}

\begin{verbatim}
| cg_add | Add |
\end{verbatim}

\textbf{Description}

Calculate $x + y$.

\textbf{Usage}

\texttt{cg\_add(x, y, name = NULL)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} either a \texttt{cg\_node} object or a numerical vector or array.
  \item \texttt{y} either a \texttt{cg\_node} object or a numerical vector or array.
  \item \texttt{name} character scalar, name of the operation (optional).
\end{itemize}

\textbf{Value}

cg\_operator object.

\textbf{Author(s)}

Ron Triepels

\textbf{See Also}

\texttt{add}
cg_asin  

**Inverse Sine**

**Description**
Calculate \( \text{asin}(x) \).

**Usage**
\[
\text{cg_asin}(x, \text{name} = \text{NULL})
\]

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

**Value**
\text{cg_operator} object.

**Author(s)**
Ron Triepels

**See Also**
- asin

---

cg_asinh  

**Inverse Hyperbolic Sine**

**Description**
Calculate \( \text{asinh}(x) \).

**Usage**
\[
\text{cg_asinh}(x, \text{name} = \text{NULL})
\]

**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.
nname 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**
```r
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**
```r
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

gc_atanh

Inverse Hyperbolic Tangent

Description

Calculate \( \text{atanh}(x) \).

Usage

\[
\text{cg_atanh}(x, \text{name} = \text{NULL})
\]

Arguments

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

Value

cg_operator object.

Author(s)

Ron Triepels

See Also

atan
cg_colmeans  

**Column Means**

**Description**

Calculate `colMeans(x)`.

**Usage**

```r
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**

```r
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

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**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate ( \cosh(x) ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{cg_cosh}(x, \text{name} = \text{NULL}) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
</tr>
<tr>
<td>( \text{name} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cg_operator object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ron Triepels</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>See Also</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \cosh )</td>
</tr>
</tbody>
</table>

### cg_crossprod

**Matrix Crossproduct**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculate ( \text{crossprod}(x, y) ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{cg_crossprod}(x, y = x, \text{name} = \text{NULL}) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
</tr>
<tr>
<td>( y )</td>
</tr>
<tr>
<td>( \text{name} )</td>
</tr>
</tbody>
</table>
**Value**

`cg_operator` object.

**Author(s)**

Ron Triepels

**See Also**

crossprod

cg_dim

---

### cg_dim

**Dimensions of an Array**

**Description**

Calculate \( \text{dim}(x) \).

**Usage**

\[
\text{cg_dim}(x, \text{name} = \text{NULL})
\]

**Arguments**

- \( x \): either a `cg_node` object or a numerical array.
- \( \text{name} \): 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
**Description**

Calculate \( x / y \).

**Usage**

\[
\text{cg\_div}(x, y, \text{name} = \text{NULL})
\]

**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

---

**Description**

Calculate \( \exp(x) \).

**Usage**

\[
\text{cg\_exp}(x, \text{name} = \text{NULL})
\]

**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 object.

Description

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

Usage

cg_function(def, grads = list())

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>def</td>
<td>function, the definition of the function.</td>
</tr>
<tr>
<td>grads</td>
<td>list of functions, the gradient functions with respect to each input (optional).</td>
</tr>
</tbody>
</table>

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**

---

**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**

---

**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 differentiated 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

```r
# 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**

```r
# 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)
```

---

**Description**

Add an input node to the active graph.

**Usage**

```r
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

```r
# 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

```r
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

**Description**

Calculate $x \times y + c(z)$.

**Usage**

```r
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

**Description**

Calculate $\log(x)$.

**Usage**

```r
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 $\log_{10}(x)$.

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 \( \log_2(x) \).

**Usage**
\[
\text{cg\_log2}(x, \text{name} = \text{NULL})
\]

**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 \( x \times y \).

**Usage**
\[
\text{cg\_matmul}(x, y, \text{name} = \text{NULL})
\]

**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 \( \text{max}(x) \).

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 \text{max} is called without changing the default value of argument \text{na.rm}.

Author(s)

Ron Triepels

See Also

max
**cg_mean**

---

### Description

Calculate mean(x).

### Usage

```r
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**

---

### Description

Calculate min(x).

### Usage

```r
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 \( x \times y \).

Usage

\[
\text{cg_mul}(x, y, \text{name} = \text{NULL})
\]

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 \( \text{n}\text{col}(x) \).

**Usage**

\[
\text{cg\_n}\text{col}(x, \text{name} = \text{NULL})
\]

**Arguments**

- **x**: either a \text{cg\_node} object or a numerical array.
- **name**: character scalar, name of the operation (optional).

**Value**

\text{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**

\text{ncol}

---

**cg_neg**

*Negative*

**Description**

Calculate \(-x\).

**Usage**

\[
\text{cg\_n}\text{eg}(x, \text{name} = \text{NULL})
\]

**Arguments**

- **x**: either a \text{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
**Description**

Add an operation node to the active graph.

**Usage**

```r
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 matched when the function is evaluated. In case no names are provided, arguments are matched positionally.

**Author(s)**

Ron Triepels

**Examples**

```r
# 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")
```
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.
nname  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 $p_{max}(x, y)$.

**Usage**

```r
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 $p_{min}(x, y)$.

**Usage**

```r
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_pow**  
*Power*

---

**Description**  
Calculate $x^y$.

**Usage**  
```r
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**  
```r
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
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`

---

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 infix 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

```r
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

sin
cg_sqrt

---

### cg_sqrt

**Square Root**

#### Description

Calculate $\sqrt{x}$.

#### Usage

```r
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

---

### cg_square

**Square**

#### Description

Calculate $x^2$.

#### Usage

```r
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

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
**Description**

Calculate $x[\ldots]$.

**Usage**

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

**Arguments**

- `x`: either a `cg_node` object or a numerical vector or array.
- `\ldots`: 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 \ldots. Any attempt to differentiate this operator with respect to these arguments results in an error.

**Author(s)**

Ron Triepels

**See Also**

- `subset`

---

**Description**

Calculate $x[[\ldots]]$.

**Usage**

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

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 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**

\[
\text{cg\_sum}(x, \text{name} = \text{NULL})
\]

**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 \text{sum} function, this function only accepts a single argument. Function \text{sum} is called without changing the default value of argument \text{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**

\[
\text{cg\_tan}(x, \text{name} = \text{NULL})
\]

**Arguments**

- **x**
  - either a cg\_node object or a numerical vector or array.
- **name**
  - character scalar, name of the operation (optional).

**Value**

\[
\text{cg\_operator object.}
\]

**Author(s)**

Ron Triepels

**See Also**

\[
\text{tan}
\]

**cg_tanh**

**Hyperbolic Tangent**

**Description**

Calculate \( \tanh(x) \).

**Usage**

\[
\text{cg\_tanh}(x, \text{name} = \text{NULL})
\]

**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

abs, 3  
acos, 4  
acosh, 5  
add, 5  
as.double, 7  
as.numeric, 8  
asin, 6  
asinh, 7  
atan, 9  
atanh, 9  
cg_abs, 3  
cg_acos, 4  
cg_acosh, 4  
cg_add, 5  
cg_as_double, 7  
cg_as_numeric, 8  
cg_asin, 6  
cg_asinh, 6  
cg_atan, 8  
cg_atanh, 9  
cg_colmeans, 10  
cg_colsums, 10  
cg_constant, 11  
cg_cos, 12  
cg_cosh, 13  
cg_crossprod, 13  
cg_dim, 14  
cg_div, 15  
cg_exp, 15  
cg_function, 16  
cg_graph, 17  
cg_graph_backward, 17  
cg_graph_forward, 18, 19  
cg_graph_get, 20  
cg_input, 21  
cg_length, 22  
cg_linear, 23  
cg_ln, 23  
cg_log10, 24  
cg_log2, 25  
cg_matmul, 25  
cg_max, 26  
cg_mean, 27  
cg_min, 27  
cg_mul, 28  
cg_ncol, 29  
cg_neg, 29  
cg_nrow, 30  
cg_operator, 31  
cg_parameter, 32  
cg_pmax, 33  
cg_pmin, 33  
cg_pos, 34  
cg_pow, 35  
cg_prod, 35  
cg_rowmeans, 36  
cg_rowsums, 37  
cg_session_get_graph, 37  
cg_session_set_graph, 38  
cg_sigmoid, 39  
cg_sin, 39  
cg_sinh, 40  
cg_sqrt, 41  
cg_square, 41  
cg_sub, 42  
cg_subset1, 43  
cg_subset2, 43  
cg_sum, 44  
cg_t, 45  
cg_tan, 46  
cg_tanh, 46  
cg_tcrossprod, 47  
colMeans, 10  
colSums, 11  
cos, 12  
cosh, 13  
crossprod, 14  
dim, 14
### INDEX

- divide, 15
- exp, 16
- length, 22
- log, 24
- log10, 24
- log2, 25
- matmult, 26
- max, 26
- mean, 27
- min, 28
- multiply, 28
- ncol, 29
- negative, 30
- nrow, 30
- pmax, 33
- pmin, 34
- positive, 34
- power, 35
- prod, 36
- rowMeans, 36
- rowSums, 37
- sin, 40
- sinh, 40
- sqrt, 41
- square, 42
- subset, 43, 44
- subtract, 42
- sum, 44, 45
- t, 45
- tan, 46
- tanh, 47
- tcrossprod, 47