

AMORE++

pre-alpha (active development aiming to release a beta version this
summer (2011))

Generated by Doxygen 1.7.4

Fri May 27 2011 02:57:19

Contents

1	The AMORE++ package	1
1.1	Introduction	1
1.2	Motivation	1
1.3	Road Map	1
2	Todo List	3
3	Class Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Class Documentation	11
6.1	Con Class Reference	11
6.1.1	Detailed Description	13
6.1.2	Member Function Documentation	13
6.1.2.1	getFromId	13
6.1.2.2	getFromNeuron	14
6.1.2.3	getWeight	15
6.1.2.4	setFromNeuron	16
6.1.2.5	setWeight	17
6.1.2.6	show	17
6.1.2.7	validate	18

6.1.3	Member Data Documentation	19
6.1.3.1	from	19
6.1.3.2	weight	19
6.2	Neuron Class Reference	19
6.2.1	Detailed Description	21
6.2.2	Member Function Documentation	21
6.2.2.1	getId	21
6.2.2.2	setId	21
6.2.3	Member Data Documentation	22
6.2.3.1	Id	22
6.2.3.2	outputValue	22
6.2.3.3	vecCon	22
6.3	vecAMORE< T > Class Template Reference	22
6.3.1	Detailed Description	24
6.3.2	Member Function Documentation	24
6.3.2.1	push_back	24
6.3.2.2	show	24
6.3.2.3	validate	24
6.3.3	Member Data Documentation	24
6.3.3.1	ldata	24
6.4	vecCon Class Reference	25
6.4.1	Detailed Description	27
6.4.2	Member Function Documentation	27
6.4.2.1	getFromId	27
6.4.2.2	numOfCons	28
7	File Documentation	29
7.1	pkg/AMORE/src/AMORE.h File Reference	29
7.2	pkg/AMORE/src/Con.cpp File Reference	30
7.3	pkg/AMORE/src/Con.h File Reference	31
7.4	pkg/AMORE/src/Neuron.cpp File Reference	31
7.5	pkg/AMORE/src/Neuron.h File Reference	33
7.6	pkg/AMORE/src/vecAMORE.cpp File Reference	33
7.7	pkg/AMORE/src/vecAMORE.h File Reference	34

7.8	pkg/AMORE/src/vecCon.cpp File Reference	34
7.9	pkg/AMORE/src/vecCon.h File Reference	34

Chapter 1

The AMORE++ package

1.1 Introduction

Here you will find the documentation of the C++ component of the AMORE++ R package. The AMORE++ package is a new version of the publicly available AMORE package for neural network training and simulation under R

1.2 Motivation

Since the release of the previous version of the AMORE many things have changed in the R programming world. The advent of the Reference Classes and of packages like Rcpp, inline and RUnit compel us to write a better version of the package in order to provide a more useful framework for neural network training and simulation.

1.3 Road Map

This project is currently very active and the development team intends to provide a beta version as soon as this summer (2011)

Chapter 2

Todo List

Member **Neuron::vecCon** restore vecCon<Con> listCon;

Chapter 3

Class Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Con	11
Neuron	19
vecAMORE< T >	22
vecAMORE< Con >	22
vecCon	25

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Con (A class to handle the information needed to describe an input connection)	11
Neuron (A class to handle the information contained in a general Neuron)	19
vecAMORE< T >	22
vecCon (A vector of connections)	25

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

pkg/AMORE/src/ AMORE.h	29
pkg/AMORE/src/ Con.cpp	30
pkg/AMORE/src/ Con.h	31
pkg/AMORE/src/ Neuron.cpp	31
pkg/AMORE/src/ Neuron.h	33
pkg/AMORE/src/ vecAMORE.cpp	33
pkg/AMORE/src/ vecAMORE.h	34
pkg/AMORE/src/ vecCon.cpp	34
pkg/AMORE/src/ vecCon.h	34

Chapter 6

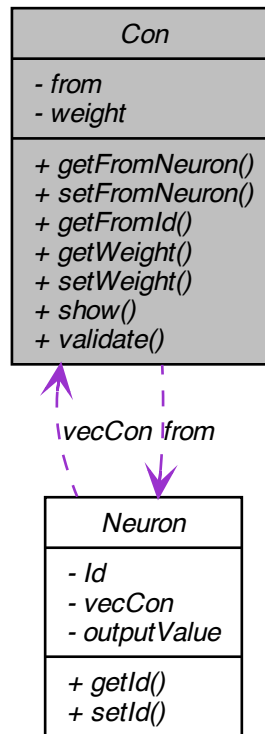
Class Documentation

6.1 Con Class Reference

A class to handle the information needed to describe an input connection.

```
#include <Con.h>
```

Collaboration diagram for Con:



Public Member Functions

- `Neuron * getFromNeuron ()`
from field accessor.
- `void setFromNeuron (Neuron *f)`
from field accessor.
- `int getFromId ()`
A getter of the Id of the Neuron pointed by the from field.
- `double getWeight ()`
weight field accessor.
- `void setWeight (double w)`
weight field accessor.
- `bool show ()`

Pretty print of the [Con](#) information.

- bool [validate](#) ()

Object validator.

Private Attributes

- [Neuron](#) * [from](#)

A pointer to the [Neuron](#) used as input during simulation or training.

- double [weight](#)

A double variable that contains the weight of the connection.

6.1.1 Detailed Description

A class to handle the information needed to describe an input connection.

The [Con](#) class provides a simple class for a connection described by a pair of values: a pointer to the [Neuron](#) used as the [from](#) field and the [weight](#) used to propagate the value of that [Neuron](#) object.

Definition at line 16 of file [Con.h](#).

6.1.2 Member Function Documentation

6.1.2.1 int [Con::getFromId](#) ()

A getter of the Id of the [Neuron](#) pointed by the [from](#) field.

This method gets the Id of the [Neuron](#) referred to by the [from](#) field

Returns

The value of the Id (an integer).

```
//=====
//Usage example:
//=====
Con myCon;
Neuron MyNeuron;
MyNeuron.setId(16);
myCon.setFromNeuron(&MyNeuron);
int result= myCon.getFromId();
```

After execution of the code shown above, [MyNeuron::Id](#) is set to the integer value 16 and, thus, result is equal to 16.

See also

[getFromNeuron](#), [setFromNeuron](#) and the unit test files, e.g., [runit.Cpp.Con.R](#), for further examples.

Definition at line 73 of file Con.cpp.

References from, and Neuron::getId().

Referenced by show(), and validate().

```

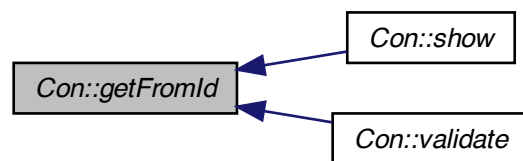
    {
        return(from->getId() );
    }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.2 Neuron * Con::getFromNeuron ()

from field accessor.

This method allows access to the address stored in the private `from` field (a pointer to a `Neuron` object).*

Returns

A pointer to the `Neuron` object referred to by the `from` field.

//=====

```
//Usage example:
//=====
Con myCon;
Neuron MyNeuron;
Neuron * ptNeuron;
MyNeuron.setId(1);
myCon.setFromNeuron(&MyNeuron);
ptNeuron = myCon.getFromNeuron();
int result= ptNeuron->getId();
```

After execution of the code shown above, ptNeuron is pointing at MyNeuron and, thus, result is equal to 1.

See also

[getFromId](#) and the unit test files, e.g., runit.Cpp.Con.R, for further examples.

Definition at line 37 of file Con.cpp.

References from.

```

{
    return(from);
}
```

6.1.2.3 double Con::getWeight ()

weight field accessor.

This method allows access to the value stored in the private field [weight](#)

Returns

The value of [weight](#) (double)

```
//=====
//Usage example:
//=====
Con myCon;
Neuron MyNeuron;
MyNeuron.setId(16);
myCon.setFromNeuron(&MyNeuron);
myCon.setWeight(12.4);
double result1= myCon.getWeight();
myCon.setWeight(2.2);
double result2= myCon.getWeight();
```

After execution of the code shown above, result1 is set to the double value 12.4 and result2 is set to the double value 2.2.

See also

[setWeight](#) and the unit test files, e.g., runit.Cpp.Con.R, for further examples.

Definition at line 103 of file Con.cpp.

References `weight`.

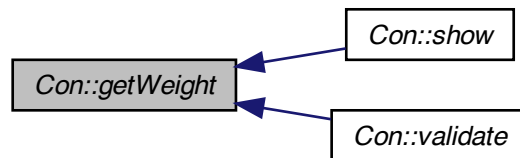
Referenced by `show()`, and `validate()`.

```

    {
        return(weight);
    }

```

Here is the caller graph for this function:



6.1.2.4 void Con::setFromNeuron (Neuron * f)

`from` field accessor.

This method sets the value of the `from` field with the address used as parameter.

Parameters

<code>f</code>	A pointer to the neuron that is to be inserted in the <code>from</code> field.
----------------	--

See also

[getFromNeuron](#) and [getFromId](#) contain usage examples. For further examples see the unit test files, e.g., `runit.Cpp.Con.R`

Definition at line 48 of file Con.cpp.

References `from`.

```

    {
        from = f;
    }

```

6.1.2.5 void Con::setWeight (double w)

weight field accessor.

This method sets the value of the [weight](#) field.

Parameters

w	The new value (double) to be set in the weight field.
----------	---

```
//=====
//Usage example:
//=====
Con myCon;
Neuron n;
n.setId(16);
myCon.setFromNeuron(&n);
myCon.setWeight(12.4);
myCon.show();
```

After execution of the code shown above, the output at the R terminal would show:

```
FROM=16 WEIGHT=12.4
```

See also

[getWeight](#) and the unit test files (e.g. `runit.Cpp.Con.R`)

Definition at line 134 of file `Con.cpp`.

References `weight`.

```

{
    weight = w;
}
```

6.1.2.6 bool Con::show ()

Pretty print of the [Con](#) information.

This method outputs in the R terminal the contents of the [Con](#) fields.

Returns

true in case everything works without throwing an exception

See also

[setWeight](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for usage examples.

Definition at line 146 of file `Con.cpp`.

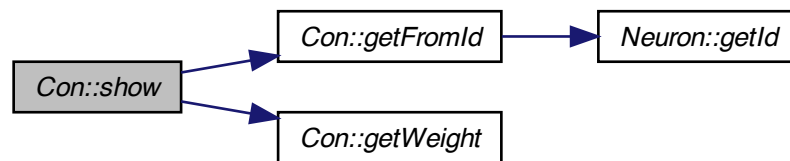
References `getFromId()`, and `getWeight()`.

```

    {
        Rprintf("From:\t %d \t Weight= \t %lf \n", getFromId() , getWeight());
        return(true);
    }

```

Here is the call graph for this function:



6.1.2.7 bool Con::validate ()

Object validator.

This method checks the object for internal coherence. A try / catch mechanism exits normal execution and returns control to the R terminal in case the contents of the [Con](#) object are identified as corrupted.

Returns

true in case the checks are Ok.

Exceptions

<i>An</i> std::range error if weight or from are not finite.
--

Definition at line 160 of file Con.cpp.

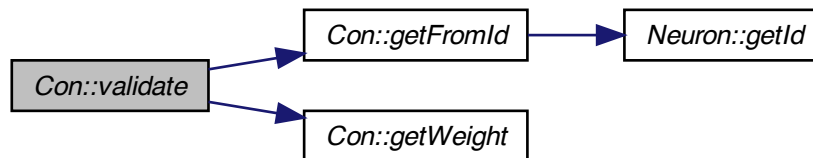
References [getFromId\(\)](#), and [getWeight\(\)](#).

```

    {
        BEGIN_RCPP
        if (! R_FINITE(getWeight()) )                throw std::range_error("weight is
not finite.");
        if (getFromId() == NA_INTEGER )              throw std::range_error("fromId is
not finite.");
        return(true);
        END_RCPP
    };

```


Here is the call graph for this function:



6.1.3 Member Data Documentation

6.1.3.1 `Neuron* Con::from` [private]

A pointer to the [Neuron](#) used as input during simulation or training.

The `from` field contains the address of the [Neuron](#) whose output will be used as input by the [Neuron](#) containing the [Con](#) object.

Definition at line 21 of file `Con.h`.

Referenced by `getFromId()`, `getFromNeuron()`, and `setFromNeuron()`.

6.1.3.2 `double Con::weight` [private]

A double variable that contains the weight of the connection.

The `weight` field contains the factor by which the output value of the [Neuron](#) addressed by the `from` field is multiplied during simulation or training.

Definition at line 26 of file `Con.h`.

Referenced by `getWeight()`, and `setWeight()`.

The documentation for this class was generated from the following files:

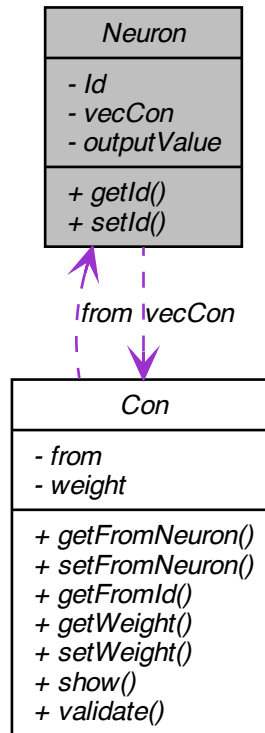
- `pkg/AMORE/src/Con.h`
- `pkg/AMORE/src/Con.cpp`

6.2 Neuron Class Reference

A class to handle the information contained in a general [Neuron](#).

```
#include <Neuron.h>
```

Collaboration diagram for Neuron:



Public Member Functions

- int `getId` ()
- void `setId` (int id)

Private Attributes

- int `Id`
An integer variable with the `Neuron` Id.
- `Con` `vecCon`
A vector of input connections.
- double `outputValue`

6.2.1 Detailed Description

A class to handle the information contained in a general [Neuron](#).

A general class for neurons. The MLPneuron and RBFneuron classes will specialize this general class

Definition at line 16 of file Neuron.h.

6.2.2 Member Function Documentation

6.2.2.1 int Neuron::getId ()

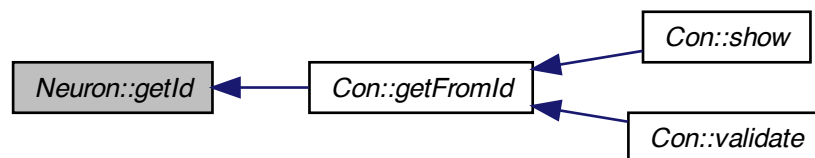
Definition at line 15 of file Neuron.cpp.

References [Id](#).

Referenced by [Con::getFromId\(\)](#).

```
    {  
        return Id;  
    }
```

Here is the caller graph for this function:



6.2.2.2 void Neuron::setId (int id)

Definition at line 19 of file Neuron.cpp.

References [Id](#).

```
    {  
        Id=id;  
    }
```

6.2.3 Member Data Documentation

6.2.3.1 `int Neuron::Id` `[private]`

An integer variable with the [Neuron](#) Id.

The [Neuron](#) Id provides a name to the neuron. This value is not expected to be used neither during simulation nor training but it provides an easy reference for human readers.

Definition at line 21 of file [Neuron.h](#).

Referenced by [getId\(\)](#), and [setId\(\)](#).

6.2.3.2 `double Neuron::outputValue` `[private]`

Definition at line 30 of file [Neuron.h](#).

6.2.3.3 `Con Neuron::vecCon` `[private]`

A vector of input connections.

[Todo](#)

```
restore vecCon<Con> listCon;
```

Definition at line 29 of file [Neuron.h](#).

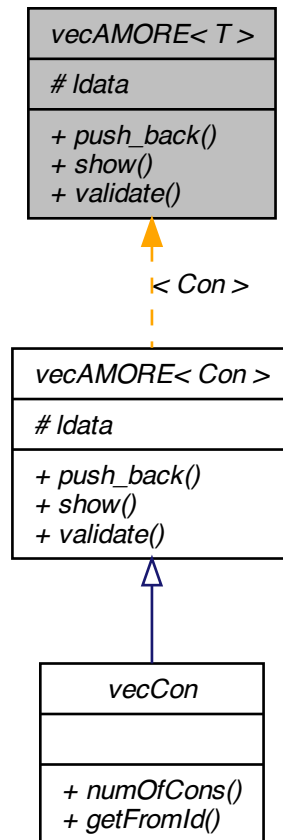
The documentation for this class was generated from the following files:

- [pkg/AMORE/src/Neuron.h](#)
- [pkg/AMORE/src/Neuron.cpp](#)

6.3 `vecAMORE< T >` Class Template Reference

```
#include <vecAMORE.h>
```

Inheritance diagram for vecAMORE< T >:



Public Member Functions

- void `push_back` (T element)
- bool `show` ()
- bool `validate` ()

Protected Attributes

- `std::vector< T > ldata`

6.3.1 Detailed Description

`template<typename T>class vecAMORE< T >`

Definition at line 11 of file vecAMORE.h.

6.3.2 Member Function Documentation

6.3.2.1 `template<typename T> void vecAMORE< T >::push_back (T element)`

Definition at line 14 of file vecAMORE.cpp.

```

{
    this->ldata.push_back(element);
};
```

6.3.2.2 `template<typename T> bool vecAMORE< T >::show ()`

Definition at line 19 of file vecAMORE.cpp.

```

{
    // for_each(ldata.begin(), ldata.end(), showCon );
    for (typename std::vector<T>::iterator itr = ldata.begin(); itr !=
ldata.end(); itr++) { itr->show(); }
    return true;
};
```

6.3.2.3 `template<typename T> bool vecAMORE< T >::validate ()`

Definition at line 26 of file vecAMORE.cpp.

```

{
    // for_each(ldata.begin(), ldata.end(), validateCon);
    for (typename std::vector<T>::iterator itr = ldata.begin(); itr !=
ldata.end(); itr++) { itr->validate(); }
    return true;
};
```

6.3.3 Member Data Documentation

6.3.3.1 `template<typename T> std::vector<T> vecAMORE< T >::ldata`
[protected]

Definition at line 13 of file vecAMORE.h.

The documentation for this class was generated from the following files:

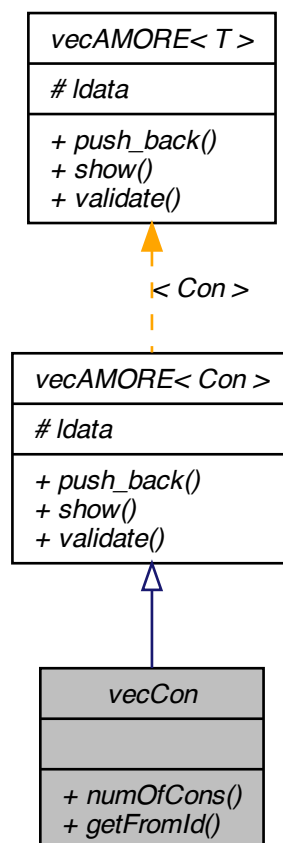
- pkg/AMORE/src/[vecAMORE.h](#)
- pkg/AMORE/src/[vecAMORE.cpp](#)

6.4 vecCon Class Reference

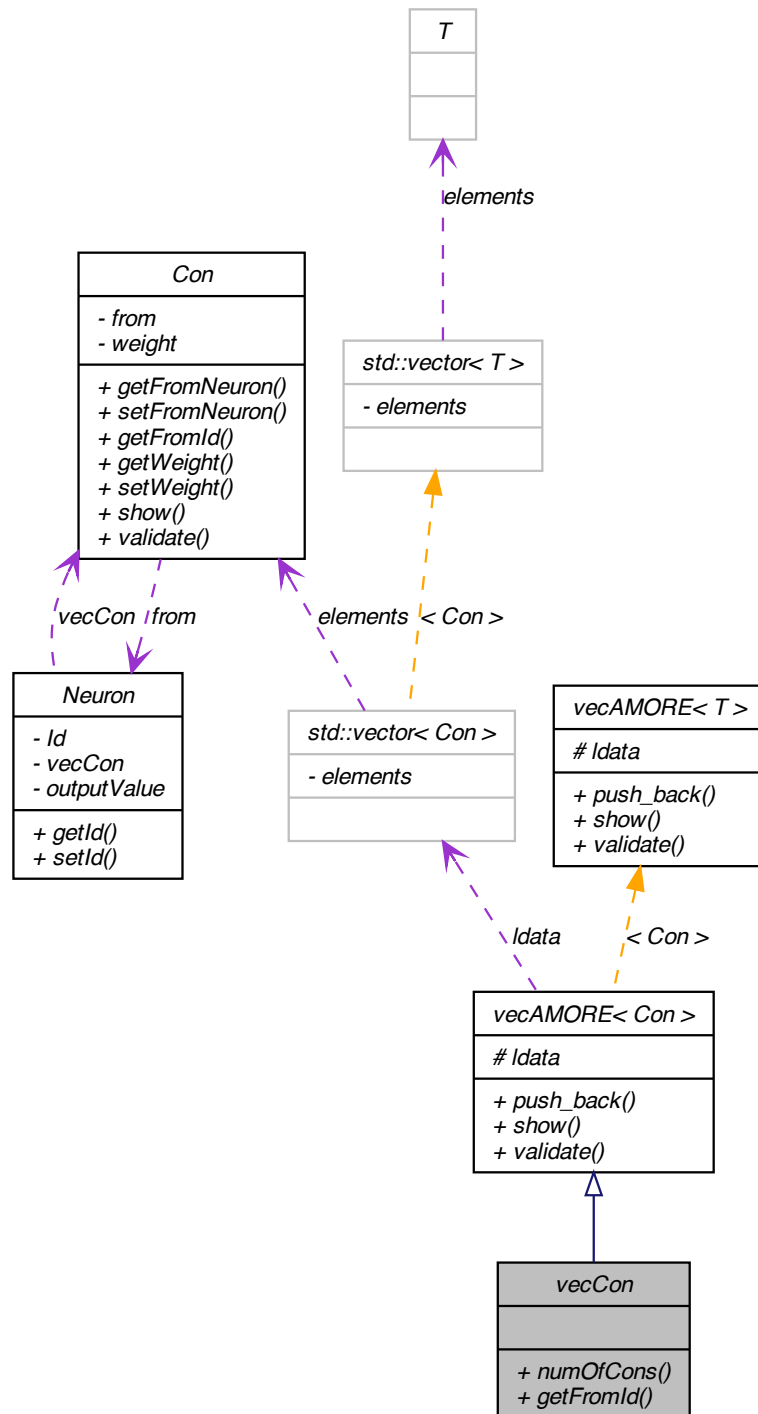
A vector of connections.

```
#include <vecCon.h>
```

Inheritance diagram for vecCon:



Collaboration diagram for vecCon:



Public Member Functions

- int `numOfCons()`
- `std::vector< int >` `getFromId()`

Getter of the Id values of the vector of Cons.

6.4.1 Detailed Description

A vector of connections.

The `vecCon` class provides a simple class for a vector of connections. It's named after the R equivalent Reference Class.

Definition at line 17 of file `vecCon.h`.

6.4.2 Member Function Documentation

6.4.2.1 `std::vector< int > vecCon::getFromId ()`

Getter of the Id values of the vector of Cons.

This function returns the Id's of the neurons referred to by the vector of Cons.

Definition at line 21 of file `vecCon.cpp`.

References `vecAMORE< Con >::ldata`, and `numOfCons()`.

```
    {  
    // for each push_back(getFromId())  
    std::vector<int> result;  
    result.reserve(numOfCons());  
    for(std::vector<Con>::iterator itr = ldata.begin();   itr != ldata.end();  
        itr++)      { result.push_back(itr->getFromId()); }  
    return result;  
}
```

Here is the call graph for this function:



6.4.2.2 int vecCon::numOfCons ()

Definition at line 11 of file vecCon.cpp.

References vecAMORE< Con >::ldata.

Referenced by getFromId().

```
        {  
        return ldata.size();  
        }
```

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- pkg/AMORE/src/[vecCon.h](#)
- pkg/AMORE/src/[vecCon.cpp](#)

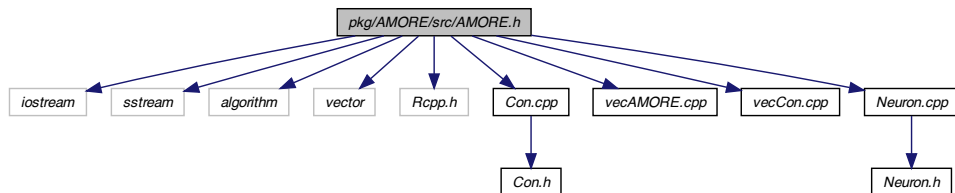
Chapter 7

File Documentation

7.1 pkg/AMORE/src/AMORE.h File Reference

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <Rcpp.h>
#include "Con.cpp"
#include "vecAMORE.cpp"
#include "vecCon.cpp"
#include "Neuron.cpp"
```

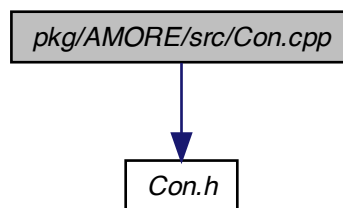
Include dependency graph for AMORE.h:



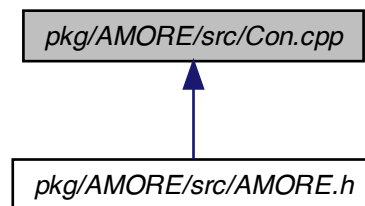
7.2 pkg/AMORE/src/Con.cpp File Reference

```
#include "Con.h"
```

Include dependency graph for Con.cpp:

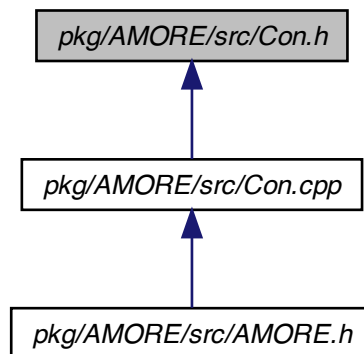


This graph shows which files directly or indirectly include this file:



7.3 pkg/AMORE/src/Con.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

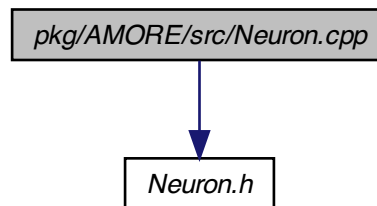
- class [Con](#)

A class to handle the information needed to describe an input connection.

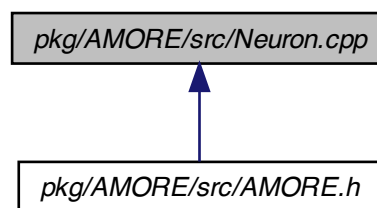
7.4 pkg/AMORE/src/Neuron.cpp File Reference

```
#include "Neuron.h"
```

Include dependency graph for Neuron.cpp:

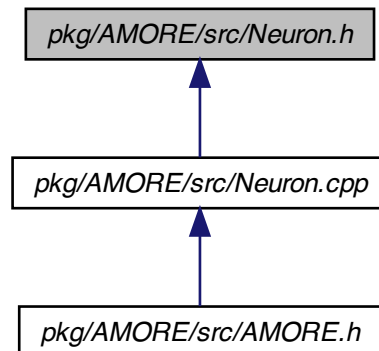


This graph shows which files directly or indirectly include this file:



7.5 pkg/AMORE/src/Neuron.h File Reference

This graph shows which files directly or indirectly include this file:

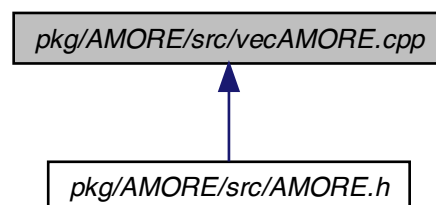


Classes

- class `Neuron`
A class to handle the information contained in a general `Neuron`.

7.6 pkg/AMORE/src/vecAMORE.cpp File Reference

This graph shows which files directly or indirectly include this file:



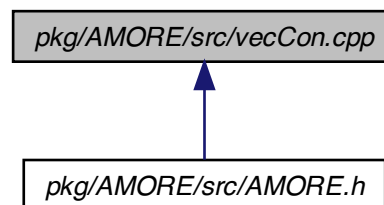
7.7 pkg/AMORE/src/vecAMORE.h File Reference

Classes

- class [vecAMORE< T >](#)

7.8 pkg/AMORE/src/vecCon.cpp File Reference

This graph shows which files directly or indirectly include this file:



7.9 pkg/AMORE/src/vecCon.h File Reference

Classes

- class [vecCon](#)
A vector of connections.

Index

- Con, [11](#)
 - from, [19](#)
 - getFromId, [13](#)
 - getFromNeuron, [14](#)
 - getWeight, [15](#)
 - setFromNeuron, [16](#)
 - setWeight, [16](#)
 - show, [17](#)
 - validate, [18](#)
 - weight, [19](#)
- from
 - Con, [19](#)
- getFromId
 - Con, [13](#)
 - vecCon, [27](#)
- getFromNeuron
 - Con, [14](#)
- getId
 - Neuron, [21](#)
- getWeight
 - Con, [15](#)
- Id
 - Neuron, [22](#)
- ldata
 - vecAMORE, [24](#)
- Neuron, [19](#)
 - getId, [21](#)
 - Id, [22](#)
 - outputValue, [22](#)
 - setId, [21](#)
 - vecCon, [22](#)
- numOfCons
 - vecCon, [27](#)
- outputValue
 - Neuron, [22](#)
- pkg/AMORE/src/AMORE.h, [29](#)
- pkg/AMORE/src/Con.cpp, [30](#)
- pkg/AMORE/src/Con.h, [31](#)
- pkg/AMORE/src/Neuron.cpp, [31](#)
- pkg/AMORE/src/Neuron.h, [33](#)
- pkg/AMORE/src/vecAMORE.cpp, [33](#)
- pkg/AMORE/src/vecAMORE.h, [34](#)
- pkg/AMORE/src/vecCon.cpp, [34](#)
- pkg/AMORE/src/vecCon.h, [34](#)
- push_back
 - vecAMORE, [24](#)
- setFromNeuron
 - Con, [16](#)
- setId
 - Neuron, [21](#)
- setWeight
 - Con, [16](#)
- show
 - Con, [17](#)
 - vecAMORE, [24](#)
- validate
 - Con, [18](#)
 - vecAMORE, [24](#)
- vecAMORE, [22](#)
 - ldata, [24](#)
 - push_back, [24](#)
 - show, [24](#)
 - validate, [24](#)
- vecCon, [25](#)
 - getFromId, [27](#)
 - Neuron, [22](#)
 - numOfCons, [27](#)
- weight
 - Con, [19](#)