CDataFrame

Generated by Doxygen 1.8.17

Chapter 1

CDataFrame: A C++ DataFrame library for Data Science and Machine Learning projects.

CDataFrame is a C++ library for Data Science and Machine Learning projects. It is designed to be fast and easy to use. It is based on the `CMatrix` library.

This library works with C++11 or higher.

1.1 Table of Contents

1.2 Installation

To install the library, follow these steps:

1. Clone the repository using the following command:

```
git clone https://github.com/B-Manitas/CDataFrame.git
```

1. Include the `CDataFrame.hpp` file in your project.

1.3 Hierarchical Structure

CMatrix is structured as follows:

Class	Description
include	
CDataFrame.hpp	The main template class that can work with any data type except bool.
src	
CDataFrame.tpp	General methods of the class.
CDataFrameConstructors.hpp	Implementation of class constructors.
CDataFrameGetter.hpp	Methods to retrieve information about the data frame and access its elements.
CDataFrameSetter.hpp	Methods to set data in the data frame.
CDataFrameCheck.tpp	Methods to verify data frame conditions and perform checks before operations to prevent errors.
CDataFrameManipulation.hpp	Methods to find elements in the data frame and transform it.
CDataFrameOperator.hpp	Implementation of various operators.
CDataFrameStatic.hpp	Implementation of static methods of the class.
test	
CDataFrameTest.hpp	Contains the tests for the class.

1.4 Documentation

For detailed information on how to use CMatrix, consult the documentation.

1.5 See Also

- CMatrix: A C++ library for matrix operations.

1.6 License

This project is licensed under the MIT License, ensuring its free and open availability to the community.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 General

Functions

- std::vector< std::string > cdata_frame< T >::__generate_uids (const size_t len, const std::string &not_in="") const

  Generate unique index.
- template<class U >
  short unsigned int cdata_frame< T >::__stream_width (const std::vector< U > data, const int &initial=0) const

  Compute the maximum length of the stream for a vector of data.
- std::vector< short unsigned int > cdata_frame< T >::__stream_widths_vec () const

  Compute the maximum length of the stream of the data.
- void cdata_frame< T >::__print_border (const std::vector< short unsigned int > &widths, const std::string &left, const std::string &middle, const std::string &right, const std::string &line="", const std::string &index="") const

  Print the border of the data frame.
- void cdata_frame< T >::__print_border_top (const std::vector< short unsigned int > &widths) const

  Print the top border of the data frame.
- void cdata_frame< T >::__print_border_middle (const std::vector< short unsigned int > &widths) const

  Print the middle border of the data frame.
- void cdata_frame< T >::__print_border_bottom (const std::vector< short unsigned int > &widths) const

  Print the bottom border of the data frame.
- template<class U >
  void cdata_frame< T >::__print_row (const std::vector< short unsigned int > &widths, const std::vector< U > &data, const std::string &index="") const

  Print a row of the data frame.
- void cdata_frame< T >::__print (std::true_type) const

  Print the data frame.
- void cdata_frame< T >::__print (std::false_type) const

  Print the data frame.
- void cdata_frame< T >::print () const

  Print the data frame.
- cdata_frame< T > cdata_frame< T >::copy () const

  Copy the data frame.
- void cdata_frame< T >::clear ()

  Cleat the data frame.

6.1.1 Detailed Description

6.1.2 Function Documentation

6.1.2.1 __generate_uids()

```
template<class T >
std::vector< std::string > cdata_frame< T >::__generate_uids (
            const size_t len,
            const std::string & not_in = "" ) const  [private]
```

Generate unique index.

Parameters

len	The length of the vector to generate.
not_in	The index to not generate. Default is "".

Returns

std::vector<std::string> The unique keys.

6.1.2.2 __print() [1/2]

```
template<class T >
void cdata_frame< T >::__print (
            std::false_type  ) const  [private]
```

Print the data frame.

Parameters

false_type	The type T is not a primitive type.

6.1.2.3 __print() [2/2]

```
template<class T >
void cdata_frame< T >::__print (
            std::true_type  ) const  [private]
```

Print the data frame.

Parameters

true_type	The type T is a primitive type.

6.1.2.4 __print_border()

```
template<class T >
void cdata_frame< T >::__print_border (
            const std::vector< short unsigned int > & widths,
            const std::string & left,
            const std::string & middle,
            const std::string & right,
            const std::string & line = "",
            const std::string & index = "" ) const  [private]
```

Print the border of the data frame.

Parameters

widths	The widths of the data frame.
left	The left border of the data frame.
middle	The middle border of the data frame.
right	The right border of the data frame.
line	The line of the data frame. Default is "".
index	The right border of the index. Default is "".

6.1.2.5 __print_border_bottom()

```
template<class T >
void cdata_frame< T >::__print_border_bottom (
            const std::vector< short unsigned int > & widths ) const  [private]
```

Print the bottom border of the data frame.

Parameters

widths	The widths of the data frame.

6.1.2.6 __print_border_middle()

```
template<class T >
void cdata_frame< T >::__print_border_middle (
            const std::vector< short unsigned int > & widths ) const  [private]
```

Print the middle border of the data frame.

Parameters

widths	The widths of the data frame.

6.1.2.7 __print_border_top()

```
template<class T >
void cdata_frame< T >::__print_border_top (
            const std::vector< short unsigned int > & widths ) const  [private]
```

Print the top border of the data frame.

Parameters

widths	The widths of the data frame.

6.1.2.8 __print_row()

```
template<class T >
template<class U >
void cdata_frame< T >::__print_row (
            const std::vector< short unsigned int > & widths,
            const std::vector< U > & data,
            const std::string & index = "" ) const  [private]
```

Print a row of the data frame.

Parameters

widths	The widths stream for each element of the row.
data	The data of the row.
index	The index of the row.

6.1.2.9 __stream_width()

```
template<class T >
template<class U >
short unsigned int cdata_frame< T >::__stream_width (
            const std::vector< U > data,
            const int & initial = 0 ) const  [private]
```

Compute the maximum length of the stream for a vector of data.

Parameters

data	The data to compute the maximum length of the stream.
initial	The initial value of the maximum length. Default is 0.

Returns

short unsigned int The maximum length of the stream of the data.

6.1.2.10 __stream_widths_vec()

```
template<class T >
std::vector< short unsigned int > cdata_frame< T >::__stream_widths_vec  [private]
```

Compute the maximum length of the stream of the data.

Returns

std::vector<short unsigned int> The maximum length of the stream of the data.

6.1.2.11 clear()

```
template<class T >
void cdata_frame< T >::clear
```

Cleat the data frame.

6.1.2.12 copy()

```
template<class T >
cdata_frame< T > cdata_frame< T >::copy
```

Copy the data frame.

Returns

cdata_frame<T> The copy of the data frame.

6.1.2.13 print()

```
template<class T >
void cdata_frame< T >::print
```

Print the data frame.

6.2 Check

Functions

- void cdata_frame< T >::__check_unique_keys (const std::string &key) const

  Check if the keys are unique.
- void cdata_frame< T >::__check_unique_index (const std::string &index) const

  Check if the index are unique.
- void cdata_frame< T >::__check_valid_row (const std::vector< T > &val) const

  Check if the row is valid.
- void cdata_frame< T >::__check_unique (const std::vector< std::string > &vec, const std::string &label) const

  Check if a vector of data is unique.
- bool cdata_frame< T >::has_keys () const

  Check if the keys are empty.
- bool cdata_frame< T >::has_index () const

  Check if the indexes are empty.

6.2.1 Detailed Description

6.2.2 Function Documentation

6.2.2.1 __check_unique()

```
template<class T >
void cdata_frame< T >::__check_unique (
            const std::vector< std::string > & vec,
            const std::string & label ) const  [private]
```

Check if a vector of data is unique.

Parameters

vec	The vector of data to check.
label	The label of the vector of data for the error message.

Exceptions

std::runtime_error	If the vector of data is not unique.

6.2.2.2 __check_unique_index()

```
template<class T >
```

```
void cdata_frame< T >::__check_unique_index (
            const std::string & index ) const  [private]
```

Check if the index are unique.

Exceptions

std::runtime_error	If the index are not unique.

6.2.2.3 __check_unique_keys()

```
template<class T >
void cdata_frame< T >::__check_unique_keys (
            const std::string & key ) const  [private]
```

Check if the keys are unique.

Exceptions

std::runtime_error	If the keys are not unique.

6.2.2.4 __check_valid_row()

```
template<class T >
void cdata_frame< T >::__check_valid_row (
            const std::vector< T > & val ) const  [private]
```

Check if the row is valid.

Parameters

val	The row to check.

Exceptions

std::invalid_argument	If the number of columns of the row is different from the number of columns of the data.

6.2.2.5 has_index()

```
template<class T >
bool cdata_frame< T >::has_index
```

Check if the indexes are empty.

Returns

true If the indexes are empty.

false If the indexes are not empty.

6.2.2.6 has_keys()

```
template<class T >
bool cdata_frame< T >::has_keys
```

Check if the keys are empty.

Returns

true If the keys are empty.

false If the keys are not empty.

6.3 Constructor

6.4 Getter

Functions

- size_t cdata_frame< T >::__get_key_pos (const std::string &key) const

  Get the position id of the key.
- size_t cdata_frame< T >::__get_index_pos (const std::string &index) const

  Get the position id of the index.
- std::vector< std::string > cdata_frame< T >::keys () const

  Get the keys.
- std::vector< std::string > cdata_frame< T >::index () const

  Get the index.
- cmatrix< T > cdata_frame< T >::data () const

  Get the data.
- cmatrix< T > cdata_frame< T >::rows (const std::string &index) const

  Get the row corresponding to the given index.
- cmatrix< T > cdata_frame< T >::rows (const std::initializer_list< std::string > &index) const

  Get the column corresponding to the given index.
- cmatrix< T > cdata_frame< T >::rows (const std::vector< std::string > &index) const

  Get the column corresponding to the given index.
- cmatrix< T > cdata_frame< T >::columns (const std::string &key) const

  Get the columns corresponding to the given keys.
- cmatrix< T > cdata_frame< T >::columns (const std::initializer_list< std::string > &keys) const

  Get the columns corresponding to the given keys.
- cmatrix< T > cdata_frame< T >::columns (const std::vector< std::string > &keys) const

  Get the columns corresponding to the given keys.

6.4.1 Detailed Description

6.4.2 Function Documentation

6.4.2.1 __get_index_pos()

```
template<class T >
size_t cdata_frame< T >::__get_index_pos (
            const std::string & index ) const  [private]
```

Get the position id of the index.

Parameters

index	The index to get the index.

Returns

size_t The position id of the index.

Exceptions

std::invalid_argument	If the index doesn't exist.

6.4.2.2 __get_key_pos()

```
template<class T >
size_t cdata_frame< T >::__get_key_pos (
            const std::string & key ) const  [private]
```

Get the position id of the key.

Parameters

key	The key to get the index.

Returns

size_t The position id of the key.

Exceptions

std::invalid_argument	If the key doesn't exist.

6.4.2.3 columns() [1/3]

```
template<class T >
cmatrix< T > cdata_frame< T >::columns (
            const std::initializer_list< std::string > & keys ) const
```

Get the columns corresponding to the given keys.

Parameters

keys	The keys of the columns to get.

Returns

cmatrix<T> The columns corresponding to the given keys.

6.4.2.4 columns() [2/3]

```
template<class T >
cmatrix< T > cdata_frame< T >::columns (
            const std::string & key ) const
```

Get the columns corresponding to the given keys.

Parameters

keys	The keys of the columns to get.

Returns

cmatrix<T> The columns corresponding to the given keys.

6.4.2.5 columns() [3/3]

```
template<class T >
cmatrix< T > cdata_frame< T >::columns (
            const std::vector< std::string > & keys ) const
```

Get the columns corresponding to the given keys.

Parameters

keys	The keys of the columns to get.

Returns

cmatrix<T> The columns corresponding to the given keys.

6.4.2.6 data()

```
template<class T >
cmatrix< T > cdata_frame< T >::data
```

Get the data.

Returns

cmatrix::CMatrix<T>

6.4.2.7 index()

```
template<class T >
std::vector< std::string > cdata_frame< T >::index
```

Get the index.

Returns

std::vector<std::string>

6.4.2.8 keys()

```
template<class T >
std::vector< std::string > cdata_frame< T >::keys
```

Get the keys.

Returns

std::vector<std::string>

6.4.2.9 rows() [1/3]

```
template<class T >
cmatrix< T > cdata_frame< T >::rows (
            const std::initializer_list< std::string > & index ) const
```

Get the column corresponding to the given index.

Parameters

index	The index of the column to get.

Returns

std::matrix<T> The column corresponding to the given index.

6.4.2.10 rows() [2/3]

```
template<class T >
cmatrix< T > cdata_frame< T >::rows (
            const std::string & index ) const
```

Get the row corresponding to the given index.

Parameters

index	The index of the row to get.

Returns

std::matrix<T> The row corresponding to the given index.

6.4.2.11 rows() [3/3]

```
template<class T >
cmatrix< T > cdata_frame< T >::rows (
            const std::vector< std::string > & index ) const
```

Get the column corresponding to the given index.

Parameters

index	The index of the column to get.

Returns

std::matrix<T> The column corresponding to the given index.

6.5 Manipulation

Functions

- void cdata_frame< T >::__remove_key (const size_t &pos)

  Remove a key at the given position.
- void cdata_frame< T >::__remove_index (const size_t &pos)

  Remove a key at the given key.

6.5.1 Detailed Description

6.5.2 Function Documentation

6.5.2.1 __remove_index()

```
template<class T >
void cdata_frame< T >::__remove_index (
            const size_t & pos )  [private]
```

Remove a key at the given key.

Parameters

key	The key of the key.

Exceptions

std::invalid_argument	If the key doesn't exist.

6.5.2.2 __remove_key()

```
template<class T >
void cdata_frame< T >::__remove_key (
            const size_t & pos )  [private]
```

Remove a key at the given position.

Parameters

pos	The position of the key.

6.6 Setter

6.7 Static

Functions

- static bool cdata_frame< T >::__is_file_exist (const std::string &path)

  Check if a file exists.
- static bool cdata_frame< T >::__has_expected_extension (const std::string &path, const std::string &extension)

  Check if a file has expected extension.
- static std::fstream cdata_frame< T >::__open_file (const std::string &path)

  Open a file.
- static std::vector< std::string > cdata_frame< T >::__parse_csv_line (const std::string &line, const char &sep, const bool &index, std::string *index_name=nullptr)

  Parse a line of a csv file.
- template<class U >
  static short unsigned int cdata_frame< T >::__count_characters (const U &input)

  Count the number of characters of a input.

6.7.1 Detailed Description

6.7.2 Function Documentation

6.7.2.1 __count_characters()

```
template<class T >
template<class U >
short unsigned int cdata_frame< T >::__count_characters (
            const U & input )  [static], [private]
```

Count the number of characters of a input.

Parameters

input	The input to count the number of characters.

Returns

short unsigned int The number of characters of the input.

6.7.2.2 __has_expected_extension()

```
template<class T >
bool cdata_frame< T >::__has_expected_extension (
```

```
            const std::string & path,
            const std::string & extension )  [static], [private]
```

Check if a file has expected extension.

Parameters

path	The path of the file.
extension	The expected extension.

Returns

true If the file has expected extension.

false If the file doesn't have expected extension.

6.7.2.3 __is_file_exist()

```
template<class T >
bool cdata_frame< T >::__is_file_exist (
            const std::string & path )  [static], [private]
```

Check if a file exists.

Parameters

path	The path of the file.

Returns

true If the file exists.

false If the file doesn't exist.

6.7.2.4 __open_file()

```
template<class T >
std::fstream cdata_frame< T >::__open_file (
            const std::string & path )  [static], [private]
```

Open a file.

Parameters

path	The path of the file.

Returns

std::fstream& The file opened.

Exceptions

std::invalid_argument	If the file doesn't exist.
std::runtime_error	If the file can't be opened.

6.7.2.5 __parse_csv_line()

```
template<class T >
std::vector< std::string > cdata_frame< T >::__parse_csv_line (
            const std::string & line,
            const char & sep,
            const bool & index,
            std::string * index_name = nullptr )  [static], [private]
```

Parse a line of a csv file.

Parameters

line	The line to parse.
sep	The separator of the csv file.
index	If the csv file has an index.
index_name	The name of the index. Default is nullptr.

Returns

std::vector<std::string> The line parsed.

Chapter 7

Class Documentation

7.1 cdata_frame< T > Class Template Reference

Main template class for the 'CDataFrame' library.

`#include <CDataFrame.hpp>`

Inheritance diagram for cdata_frame< T >:

Collaboration diagram for cdata_frame< T >:

Public Member Functions

- cdata_frame ()
- cdata_frame (const cmatrix< T > &data)
- cdata_frame (const std::vector< std::string > &keys, const cmatrix< T > &data)
- cdata_frame (const cmatrix< T > &data, const std::vector< std::string > &index)
- cdata_frame (const std::vector< std::string > &keys, const cmatrix< T > &data, const std::vector< std::string > &index)
- ~cdata_frame ()

  Destroy the CDataFrame object.
- std::vector< std::string > keys () const

  Get the keys.
- std::vector< std::string > index () const

  Get the index.
- cmatrix< T > data () const

  Get the data.
- cmatrix< T > rows (const std::string &index) const

  Get the row corresponding to the given index.
- cmatrix< T > rows (const std::initializer_list< std::string > &index) const

  Get the column corresponding to the given index.
- cmatrix< T > rows (const std::vector< std::string > &index) const

  Get the column corresponding to the given index.
- cmatrix< T > columns (const std::string &key) const

  Get the columns corresponding to the given keys.
- cmatrix< T > columns (const std::initializer_list< std::string > &keys) const

Get the columns corresponding to the given keys.

- cmatrix< T > columns (const std::vector< std::string > &keys) const

  Get the columns corresponding to the given keys.
- void set_keys (const std::vector< std::string > &keys)
- void set_index (const std::vector< std::string > &index)
- void set_data (const cmatrix< T > &data)
- void insert_row (const size_t &pos, const std::vector< T > &val, const std::string &index="")
- void insert_column (const size_t &pos, const std::vector< T > &val, const std::string &key="")
- void concatenate (const cdata_frame< T > &df, const short unsigned int &axis=0)
- void push_row_front (const std::vector< T > &val, const std::string &index="")
- void push_row_back (const std::vector< T > &val, const std::string &index="")
- void push_col_front (const std::vector< T > &val, const std::string &key="")
- void push_col_back (const std::vector< T > &val, const std::string &key="")
- void remove_row (const size_t &pos)
- void remove_row (const std::string &index)
- void remove_column (const size_t &pos)
- void remove_column (const std::string &key)
- bool has_keys () const

  Check if the keys are empty.
- bool has_index () const

  Check if the indexes are empty.
- void print () const

  Print the data frame.
- cdata_frame< T > copy () const

  Copy the data frame.
- void clear ()

  Cleat the data frame.
- bool operator== (const cdata_frame< T > &df) const

  The equality operator.
- bool operator!= (const cdata_frame< T > &df) const

  The inequality operator.

Static Public Member Functions

- static cdata_frame< std::string > read_csv (const std::string &path, const bool &header=true, const bool &index=false, const char &sep=',')
- static cdata_frame< T > merge (const cdata_frame< T > &df1, const cdata_frame< T > &df2, const unsigned int &axis=0)

Private Member Functions

- size_t __get_key_pos (const std::string &key) const

  Get the position id of the key.
- size_t __get_index_pos (const std::string &index) const

  Get the position id of the index.
- void __remove_key (const size_t &pos)

  Remove a key at the given position.
- void __remove_index (const size_t &pos)

  Remove a key at the given key.
- std::vector< std::string > __generate_uids (const size_t len, const std::string &not_in="") const

  Generate unique index.

- template<class U >
  short unsigned int __stream_width (const std::vector< U > data, const int &initial=0) const

  Compute the maximum length of the stream for a vector of data.
- std::vector< short unsigned int > __stream_widths_vec () const

  Compute the maximum length of the stream of the data.
- void __print_border (const std::vector< short unsigned int > &widths, const std::string &left, const std::string &middle, const std::string &right, const std::string &line="", const std::string &index="") const

  Print the border of the data frame.
- void __print_border_top (const std::vector< short unsigned int > &widths) const

  Print the top border of the data frame.
- void __print_border_middle (const std::vector< short unsigned int > &widths) const

  Print the middle border of the data frame.
- void __print_border_bottom (const std::vector< short unsigned int > &widths) const

  Print the bottom border of the data frame.
- template<class U >
  void __print_row (const std::vector< short unsigned int > &widths, const std::vector< U > &data, const std::string &index="") const

  Print a row of the data frame.
- void __print (std::true_type) const

  Print the data frame.
- void __print (std::false_type) const

  Print the data frame.
- void __check_unique_keys (const std::string &key) const

  Check if the keys are unique.
- void __check_unique_index (const std::string &index) const

  Check if the index are unique.
- void __check_valid_row (const std::vector< T > &val) const

  Check if the row is valid.
- void __check_unique (const std::vector< std::string > &vec, const std::string &label) const

  Check if a vector of data is unique.

Static Private Member Functions

- static bool __is_file_exist (const std::string &path)

  Check if a file exists.
- static bool __has_expected_extension (const std::string &path, const std::string &extension)

  Check if a file has expected extension.
- static std::fstream __open_file (const std::string &path)

  Open a file.
- static std::vector< std::string > __parse_csv_line (const std::string &line, const char &sep, const bool &index, std::string *index_name=nullptr)

  Parse a line of a csv file.
- template<class U >
  static short unsigned int __count_characters (const U &input)

  Count the number of characters of a input.

Private Attributes

- std::vector< std::string > m_keys = std::vector<std::string>()
- std::vector< std::string > m_index = std::vector<std::string>()

7.1.1 Detailed Description

template<typename T>
class cdata_frame< T >

Main template class for the 'CDataFrame' library.

Template Parameters

T	The type of the data.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 cdata_frame() [1/5]

```
template<class T >
cdata_frame< T >::cdata_frame
```

7.1.2.2 cdata_frame() [2/5]

```
template<class T >
cdata_frame< T >::cdata_frame (
            const cmatrix< T > & data )
```

7.1.2.3 cdata_frame() [3/5]

```
template<class T >
cdata_frame< T >::cdata_frame (
            const std::vector< std::string > & keys,
            const cmatrix< T > & data )
```

7.1.2.4 cdata_frame() [4/5]

```
template<class T >
cdata_frame< T >::cdata_frame (
            const cmatrix< T > & data,
            const std::vector< std::string > & index )
```

7.1.2.5 cdata_frame() [5/5]

```
template<class T >
cdata_frame< T >::cdata_frame (
            const std::vector< std::string > & keys,
            const cmatrix< T > & data,
            const std::vector< std::string > & index )
```

7.1.2.6 ~cdata_frame()

```
template<class T >
cdata_frame< T >::~cdata_frame
```

Destroy the CDataFrame object.

7.1.3 Member Function Documentation

7.1.3.1 concatenate()

```
template<class T >
void cdata_frame< T >::concatenate (
            const cdata_frame< T > & df,
            const short unsigned int & axis = 0 )
```

7.1.3.2 insert_column()

```
template<class T >
void cdata_frame< T >::insert_column (
            const size_t & pos,
            const std::vector< T > & val,
            const std::string & key = "" )
```

7.1.3.3 insert_row()

```
template<class T >
void cdata_frame< T >::insert_row (
            const size_t & pos,
            const std::vector< T > & val,
            const std::string & index = "" )
```

7.1.3.4 merge()

```
template<class T >
cdata_frame< T > cdata_frame< T >::merge (
            const cdata_frame< T > & df1,
            const cdata_frame< T > & df2,
            const unsigned int & axis = 0 )  [static]
```

7.1.3.5 operator"!=()

```
template<class T >
bool cdata_frame< T >::operator!= (
            const cdata_frame< T > & df ) const
```

The inequality operator.

Parameters

df	The data frame to compare.

Returns

true If the data frames are not equal.

false If the data frames are equal.

7.1.3.6 operator==()

```
template<class T >
bool cdata_frame< T >::operator== (
            const cdata_frame< T > & df ) const
```

The equality operator.

Parameters

df	The data frame to compare.

Returns

true If the data frames are equal.

false If the data frames are not equal.

7.1.3.7 push_col_back()

```
template<class T >
void cdata_frame< T >::push_col_back (
            const std::vector< T > & val,
            const std::string & key = "" )
```

7.1.3.8 push_col_front()

```
template<class T >
void cdata_frame< T >::push_col_front (
            const std::vector< T > & val,
            const std::string & key = "" )
```

7.1.3.9 push_row_back()

```
template<class T >
void cdata_frame< T >::push_row_back (
            const std::vector< T > & val,
            const std::string & index = "" )
```

7.1.3.10 push_row_front()

```
template<class T >
void cdata_frame< T >::push_row_front (
            const std::vector< T > & val,
            const std::string & index = "" )
```

7.1.3.11 read_csv()

```
template<class T >
cdata_frame< std::string > cdata_frame< T >::read_csv (
            const std::string & path,
            const bool & header = true,
            const bool & index = false,
            const char & sep = ',' )  [static]
```

7.1.3.12 remove_column() [1/2]

```
template<class T >
void cdata_frame< T >::remove_column (
            const size_t & pos )
```

7.1.3.13 remove_column() [2/2]

```
template<class T >
void cdata_frame< T >::remove_column (
            const std::string & key )
```

7.1.3.14 remove_row() [1/2]

```
template<class T >
void cdata_frame< T >::remove_row (
            const size_t & pos )
```

7.1.3.15 remove_row() [2/2]

```
template<class T >
void cdata_frame< T >::remove_row (
            const std::string & index )
```

7.1.3.16 set_data()

```
template<class T >
void cdata_frame< T >::set_data (
            const cmatrix< T > & data )
```

7.1.3.17 set_index()

```
template<class T >
void cdata_frame< T >::set_index (
            const std::vector< std::string > & index )
```

7.1.3.18 set_keys()

```
template<class T >
void cdata_frame< T >::set_keys (
            const std::vector< std::string > & keys )
```

7.1.4 Member Data Documentation

7.1.4.1 m_index

```
template<typename T >
std::vector<std::string> cdata_frame< T >::m_index = std::vector<std::string>()  [private]
```

7.1.4.2 m_keys

```
template<typename T >
std::vector<std::string> cdata_frame< T >::m_keys = std::vector<std::string>()  [private]
```

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

- include/CDataFrame.hpp
- src/CDataFrame.tpp
- src/CDataFrameCheck.tpp
- src/CDataFrameConstructor.tpp
- src/CDataFrameGetter.tpp
- src/CDataFrameManipulation.tpp
- src/CDataFrameOperator.tpp
- src/CDataFrameSetter.tpp
- src/CDataFrameStatic.tpp

Chapter 8

File Documentation

8.1 include/CDataFrame.hpp File Reference

File containing the main template class for the 'CDataFrame' library.

```
#include <fstream>
#include <initializer_list>
#include <set>
#include <sstream>
#include <string>
#include <tuple>
#include <vector>
#include "../lib/CMatrix/include/CMatrix.hpp"
#include "../src/CDataFrameCheck.tpp"
#include "../src/CDataFrameConstructor.tpp"
#include "../src/CDataFrameGetter.tpp"
#include "../src/CDataFrameManipulation.tpp"
#include "../src/CDataFrameOperator.tpp"
#include "../src/CDataFrameSetter.tpp"
#include "../src/CDataFrameStatic.tpp"
#include "../src/CDataFrame.tpp"
```

Include dependency graph for CDataFrame.hpp:

8.2 readme.md File Reference

8.3 src/CDataFrame.tpp File Reference

File containing the general methods of the 'DataFrame' class.

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

8.3.1 Detailed Description

File containing the general methods of the 'DataFrame' class.

This file contains the implementation of operators and methods of the 'CDataFrame' class.

See also

CDataFrame.hpp

8.4 src/CDataFrameCheck.tpp File Reference

File containing the implementation of check methods of the 'DataFrame' class.

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

8.4.1 Detailed Description

File containing the implementation of check methods of the 'DataFrame' class.

See also

CDataFrame.hpp

8.5 src/CDataFrameConstructor.tpp File Reference

File containing the constructor and destructor of the 'DataFrame' class.

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

8.5.1 Detailed Description

File containing the constructor and destructor of the 'DataFrame' class.

See also

CDataFrame.hpp

8.6 src/CDataFrameGetter.tpp File Reference

File containing the implementation of getters of the 'DataFrame' class.

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

8.6.1 Detailed Description

File containing the implementation of getters of the 'DataFrame' class.

See also

CDataFrame.hpp

8.7 src/CDataFrameManipulation.tpp File Reference

File containing the implementation of manipulation methods of the 'DataFrame' class.

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

8.7.1 Detailed Description

File containing the implementation of manipulation methods of the 'DataFrame' class.

See also

CDataFrame.hpp

8.8 src/CDataFrameOperator.tpp File Reference

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

8.9 src/CDataFrameSetter.tpp File Reference

File containing the implementation of setters of the 'DataFrame' class.

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

8.9.1 Detailed Description

File containing the implementation of setters of the 'DataFrame' class.

See also

CDataFrame.hpp

8.10 src/CDataFrameStatic.tpp File Reference

File containing the implementation of static methods of the 'DataFrame' class.

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

8.10.1 Detailed Description

File containing the implementation of static methods of the 'DataFrame' class.

See also

CDataFrame.hpp

Chapter 9

Example Documentation

9.1 /home/manitas/Documents/Programming/Data Science/ETL/CDataFrame/include/CDataFrame.hpp

Construct a new CDataFrame object.

Note

The data, keys and index are empty.

cdata_frame<int> df = cdata_frame<int>();

Science/ETL/CDataFrame/include/CDataFrame.hpp