Data Frame

0.1

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 DF Namespace Reference

Classes

- class DataFrame

  DataFrame is class for parsing data in a given file with a give delimeter (default is comma ','). all data will be saved as string wich user can later covert to desired type.

4.2 std Namespace Reference

Namespaces

- shorts

  namespace for introducting shortnames

4.3 std::shorts Namespace Reference

namespace for introducting shortnames

Typedefs

- using V_string = vector< string >
- using VV_string = vector< V_string >
- using V_any = vector< any >
- using VV_any = vector< vector< any > >
- using Data = unordered_map< string, V_string >
- using V_double = vector< double >
- using V_int = vector< int >
- using V_pair_ints = std::vector< std::pair< int, int > >

4.3.1 Detailed Description

namespace for introducting shortnames

4.3.2 Typedef Documentation

4.3.2.1 Data

```
using std::shorts::Data = typedef unordered_map<string, V_string>
```

Definition at line 39 of file ReadFiles.hpp.

4.3.2.2 V_any

```
using std::shorts::V_any = typedef vector<any>
```

Definition at line 37 of file ReadFiles.hpp.

4.3.2.3 V_double

```
using std::shorts::V_double = typedef vector<double>
```

Definition at line 40 of file ReadFiles.hpp.

4.3.2.4 V_int

```
using std::shorts::V_int = typedef vector<int>
```

Definition at line 41 of file ReadFiles.hpp.

4.3.2.5 V_pair_ints

```
using std::shorts::V_pair_ints = typedef std::vector<std::pair<int, int> >
```

Definition at line 42 of file ReadFiles.hpp.

4.3.2.6 V_string

```
using std::shorts::V_string = typedef vector<string>
```

Definition at line 35 of file ReadFiles.hpp.

4.3.2.7 VV_any

```
using std::shorts::VV_any = typedef vector<vector<any> >
```

Definition at line 38 of file ReadFiles.hpp.

4.3.2.8 VV_string

```
using std::shorts::VV_string = typedef vector<V_string>
```

Definition at line 36 of file ReadFiles.hpp.

Chapter 5

Class Documentation

5.1 DF::DataFrame Class Reference

DataFrame is class for parsing data in a given file with a give delimeter (default is comma ','). all data will be saved as string wich user can later covert to desired type.

```
#include <ReadFiles.hpp>
```

Public Member Functions

- int get_n_rows () const

  Get the number of rows of a given data.
- int get_n_cols () const

  Get the number cols of a given data.
- void set_headers (std::shorts::V_string const &v_hdrs)

  Set the headers with user provided vector of strings.
- std::shorts::V_string get_headers () const

  Get the headers.
- std::shorts::V_string get_by_header (std::string const &hdr)
- DataFrame copy () const

  to copy current data into new data frame
- DataFrame copy_by_headers (std::shorts::V_string const &v_hdrs)

  copy the requested data by provided headers name as vector of strings into a new data frame
- void remove_duplications (std::string const &hdr)

  get a given header and only keep the first occurance and remove the remaning rows
- void read_files (std::string_view path, char delim=',', bool is_first_col_header=true, std::shorts::V_string v_↩
hdrs={})

  read files
- void read_text (std::string const &text, std::shorts::V_pair_ints const &v_cols_start_length, bool is_first_col↩
_header=true, std::shorts::V_string v_hdrs={})
- void head (unsigned long long n=5)

  print n first rows off all columns
- void swap_cols_pos (std::string first_hdr, std::string second_hdr)

  swap two columns with their header with each others

Public Attributes

- unsigned long long n_rows

  number of rows
- unsigned long long n_cols

  number of columns
- std::unordered_map< int, int > mising_values

  an unorderd maps for missing values

Private Member Functions

- std::shorts::V_string read_lines (std::string_view path)
- std::shorts::V_string read_lines (std::string const &text)
- std::shorts::V_string parse_line (std::string const &line, char delim)
- std::shorts::V_string parse_line (std::string const &line, std::shorts::V_pair_ints const &v_cols_start_ends)
- void fill_data (std::shorts::V_string const &v_lines, char delim=',', bool is_first_col_header=true, std::shorts::V_string v_hdrs={})
- void fill_data (std::shorts::V_string const &v_lines, std::shorts::V_pair_ints const &v_cols_start_length, bool is_first_col_header=true, std::shorts::V_string v_hdrs={})

Private Attributes

- std::shorts::Data data
- std::shorts::V_string headers

5.1.1 Detailed Description

DataFrame is class for parsing data in a given file with a give delimeter (default is comma ','). all data will be saved as string wich user can later covert to desired type.

Definition at line 54 of file ReadFiles.hpp.

5.1.2 Member Function Documentation

5.1.2.1 copy()

```
DataFrame DF::DataFrame::copy ( ) const
```

to copy current data into new data frame

Returns

DataFrame new data frame

5.1.2.2 copy_by_headers()

```
DF::DataFrame DF::DataFrame::copy_by_headers (
            std::shorts::V_string const & v_hdrs )
```

copy the requested data by provided headers name as vector of strings into a new data frame

Parameters

v_hdrs	

Returns

DataFrame new data frame

Definition at line 300 of file ReadFiles.cpp.

```
{
    DF::DataFrame new_df;
    new_df.n_cols = v_hdrs.size();
    new_df.n_rows = n_rows;
    new_df.headers = v_hdrs;

    for(auto const& hdr : v_hdrs)
    {
        new_df.data[hdr] = data[hdr];
    }

    return new_df;
}
```

5.1.2.3 fill_data() [1/2]

```
void DF::DataFrame::fill_data (
            std::shorts::V_string const & v_lines,
            char delim = ',',
            bool is_first_col_header = true,
            std::shorts::V_string v_hdrs = {} )  [private]
```

Definition at line 97 of file ReadFiles.cpp.

```
{
   std::shorts::VV_string vv_strs;

    for(auto const& line : lines)
    {
        std::shorts::V_string v_str_tmp = parse_line(line, delim);
        vv_strs.emplace_back(v_str_tmp);
    }

    // init n_rows and n_cols
    n_rows = vv_strs.size();
    n_cols = vv_strs[0].size();

    headers.resize(n_cols);

    // initializing headers
    for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
    {
        if(is_first_col_header)
        {
            headers[i_col] = vv_strs[0][i_col];
        }
        else if(v_hdrs.size() > 0)
        {
            if(v_hdrs.size() == n_cols)
            {
                headers[i_col] = v_hdrs[i_col];
            }
            else
            {
                throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: number of provided
   headers does not match with the number of columns in the data"));
            }
        }
        else
        {
            headers[i_col] = std::to_string(i_col + 1);
        }
    }

    for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
    {
        std::shorts::V_string values;
        for(unsigned long long i_row{is_first_col_header}; i_row < n_rows; ++i_row)
        {
            if(vv_strs[i_row].size() != n_cols)
            {
                throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: inconsistent number of
       columns, check row {}", i_row + 1));
            }

            values.emplace_back(vv_strs[i_row][i_col]);

            // check for the missig values
            // missing values are empty string, NA, and NAN
            if(vv_strs[i_row][i_col] == ""    ||
               vv_strs[i_row][i_col] == "NA"  ||
               vv_strs[i_row][i_col] == "NAN")
            {
                mising_values.insert({i_row, i_col});
            }
        }

        data[headers[i_col]] = values;
    }
}
```

5.1.2.4 fill_data() [2/2]

```
void DF::DataFrame::fill_data (
            std::shorts::V_string const & v_lines,
            std::shorts::V_pair_ints const & v_cols_start_length,
            bool is_first_col_header = true,
            std::shorts::V_string v_hdrs = {} )   [private]
```

Definition at line 163 of file ReadFiles.cpp.

```
{
    std::shorts::VV_string vv_strs;
    for(auto const& line : lines)
    {
        auto v_str_tmp = parse_line(line, v_cols_start_length);
        vv_strs.emplace_back(v_str_tmp);
    }

    // init n_rows and n_cols
    n_rows = vv_strs.size();
    n_cols = vv_strs[0].size();

    headers.resize(n_cols);

    // initializing headers
    for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
    {
        if(is_first_col_header)
        {
            headers[i_col] = vv_strs[0][i_col];
        }
        else if(v_hdrs.size() > 0)
        {
            if(v_hdrs.size() == n_cols)
            {
                headers[i_col] = v_hdrs[i_col];
            }
            else
            {
                throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: number of provided
       headers does not match with the number of columns in the data"));
            }
        }
        else
        {
            headers[i_col] = std::to_string(i_col + 1);
        }
    }

    for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
    {
        std::shorts::V_string values;
        for(unsigned long long i_row{is_first_col_header}; i_row < n_rows; ++i_row)
        {
            if(vv_strs[i_row].size() != n_cols)
            {
                throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: inconsistent number of
       columns, check row {}", i_row + 1));
            }

            values.emplace_back(vv_strs[i_row][i_col]);

            // check for the missig values
            // missing values are empty string, NA, and NAN
            if(vv_strs[i_row][i_col].empty() ||
               vv_strs[i_row][i_col] == " "  ||
               vv_strs[i_row][i_col] == "NA" ||
               vv_strs[i_row][i_col] == "NAN")
            {
                mising_values.insert({i_row, i_col});
            }
        }

        data[headers[i_col]] = values;
    }
}
```

5.1.2.5 get_by_header()

```
std::shorts::V_string DF::DataFrame::get_by_header (
              std::string const & hdr )
```

Definition at line 295 of file ReadFiles.cpp.

```
{
    return data[hdr];
}
```

5.1.2.6 get_headers()

```
std::shorts::V_string DF::DataFrame::get_headers ( ) const
```

Get the headers.

Returns

std::shorts::V_string headers

Definition at line 290 of file ReadFiles.cpp.

```
{
    return headers;
}
```

5.1.2.7 get_n_cols()

```
int DF::DataFrame::get_n_cols ( ) const
```

Get the number cols of a given data.

Returns

int number of columns

Definition at line 12 of file ReadFiles.cpp.

```
{
    return n_cols;
}
```

5.1.2.8 get_n_rows()

```
int DF::DataFrame::get_n_rows ( ) const
```

Get the number of rows of a given data.

Returns

int number of rows

Definition at line 17 of file ReadFiles.cpp.

```
{
    return n_rows;
}
```

5.1.2.9 head()

```
void DF::DataFrame::head (
            unsigned long long n = 5 )
```

print n first rows off all columns

Parameters

n	

Definition at line 241 of file ReadFiles.cpp.

```
{
    if(n > n_rows)
    {
        fmt::print(fg(fmt::color::yellow),"Warning: number of row requested ({}) to be printed is bigger than number of available rows in data ({})\n",
        n, data[headers[0]].size());
        if(n_rows > 5) n = 5;
        else n = n_rows;
    }

    for(auto const& curr_hdr : headers)
    {
        fmt::print(fmt::emphasis::bold | fg(fmt::color::green), "{:^15}|", curr_hdr);
    }

    std::string sub_separator = std::string(15,'-') + '+';
    std::string separator = "";

    for(size_t i{0}; i < n_cols; ++i )
    {
        separator += sub_separator;
    }
    // fmt::println("\n{}",std::string(headers.size() * 16, '-'));
    fmt::println("\n{}",std::string(separator));

    // std::cout « '\n';

    for(unsigned long long i{0}; i < n; ++i)
    {
        for(auto const& curr_hdr : headers)
        {
            if(data[curr_hdr][i] == ""    ||
               data[curr_hdr][i] == "NA" ||
               data[curr_hdr][i] == "NAN")
            {
                fmt::print(bg(fmt::color::red),"{:^15}", data[curr_hdr][i] /*data[curr_hdr][i]*/);
                std::cout « "|";
            }
            else
            {
                fmt::print("{:^15}|", data[curr_hdr][i] /*data[curr_hdr][i]*/);
            }

        }

        std::cout « "\n";
    }
}
```

5.1.2.10 parse_line() [1/2]

```
std::shorts::V_string DF::DataFrame::parse_line (
              std::string const & line,
              char delim )   [private]
```

Definition at line 65 of file ReadFiles.cpp.

```
{
    std::shorts::V_string v_strs;

    std::istringstream iss(line);
    std::string cell;

    while(std::getline(iss, cell, delim))
    {
        cell.erase(std::remove(cell.begin(), cell.end(), '\"'), cell.end());
        cell.erase(std::remove(cell.begin(), cell.end(), ' '), cell.end());
        v_strs.emplace_back(cell);
    }

    return v_strs;
}
```

5.1.2.11 parse_line() [2/2]

```
std::shorts::V_string DF::DataFrame::parse_line (
              std::string const & line,
              std::shorts::V_pair_ints const & v_cols_start_ends )   [private]
```

Definition at line 82 of file ReadFiles.cpp.

```
{
    std::shorts::V_string v_strs;

    for(auto const& [start, end] : v_cols_start_ends)
    {
        auto cell = line.substr(start, end);
        cell.erase(std::remove(cell.begin(), cell.end(), '\"'), cell.end());
        cell.erase(std::remove(cell.begin(), cell.end(), ' '), cell.end());
        v_strs.push_back(cell);
    }

    return v_strs;
}
```

5.1.2.12 read_files()

```
void DF::DataFrame::read_files (
             std::string_view path,
             char delim = ',',
             bool is_first_col_header = true,
             std::shorts::V_string v_hdrs = {} )
```

read files

Parameters

path	path to input file
delim	delimiter for parsing the input file
is_first_col_header	boolean

Definition at line 229 of file ReadFiles.cpp.

```
{
    auto lines = read_lines(path);
    fill_data(lines, delim, is_first_col_header, v_hdrs);
}
```

5.1.2.13 read_lines() [1/2]

```
std::shorts::V_string DF::DataFrame::read_lines (
             std::string const & text )  [private]
```

Definition at line 46 of file ReadFiles.cpp.

```
{
    std::istringstream iss(text);

    std::string line;
    std::shorts::V_string v_strs;


    while(std::getline(iss, line))
    {
        if(line.size() == 0) continue;
        line.erase(std::remove(line.begin(), line.end(), '\r'),line.end());
        v_strs.emplace_back(line);
    }

    return v_strs;
}
```

5.1.2.14 read_lines() [2/2]

```
std::shorts::V_string DF::DataFrame::read_lines (
             std::string_view path )  [private]
```

Definition at line 22 of file ReadFiles.cpp.

```
{
    std::ifstream ifs(path.data());

    if(ifs.fail())
    {
        throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: unable to read file {}.\nPlease
    check your input.", path));
    }


    std::string line;
    std::shorts::V_string v_strs;


    while(std::getline(ifs, line))
    {
        if(line.size() == 0) continue;
        line.erase(std::remove(line.begin(), line.end(), '\r'),line.end());
        v_strs.emplace_back(line);
    }

    return v_strs;
}
```

5.1.2.15 read_text()

```
void DF::DataFrame::read_text (
             std::string const & text,
             std::shorts::V_pair_ints const & v_cols_start_length,
             bool is_first_col_header = true,
             std::shorts::V_string v_hdrs = {} )
```

Parameters

path	std::string_view
v_cols_length	
is_first_col_header	

Definition at line 235 of file ReadFiles.cpp.

```
{
    auto lines = read_lines(text);
    fill_data(lines, v_cols_start_length, is_first_col_header, v_hdrs);
}
```

5.1.2.16 remove_duplications()

```
void DF::DataFrame::remove_duplications (
             std::string const & hdr )
```

get a given header and only keep the first occurance and remove the remaning rows

Parameters

hdr	header to check the duplication

5.1.2.17 set_headers()

```
void DF::DataFrame::set_headers (
            std::shorts::V_string const & v_hdrs )
```

Set the headers with user provided vector of strings.

Parameters

v_hdrs	provided headers from users

5.1.2.18 swap_cols_pos()

```
void DF::DataFrame::swap_cols_pos (
            std::string first_hdr,
            std::string second_hdr )
```

swap two columns with their header with each others

Parameters

first_hdr	std::string
second_hdr	std::string

Definition at line 315 of file ReadFiles.cpp.

```
{
    std::string tmp_hdr;
    std::shorts::V_string tmp_vals;
    auto first_it = std::find(headers.begin(), headers.end(), first_hdr);
    auto second_it = std::find(headers.begin(), headers.end(), second_hdr);
    swap(headers[first_it - headers.begin()], headers[second_it - headers.begin()]);

    std::swap(data[first_hdr], data[second_hdr]);
}
```

5.1.3 Member Data Documentation

5.1.3.1 data

```
std::shorts::Data DF::DataFrame::data  [private]
```

Definition at line 163 of file ReadFiles.hpp.

5.1.3.2 headers

```
std::shorts::V_string DF::DataFrame::headers  [private]
```

Definition at line 164 of file ReadFiles.hpp.

5.1.3.3 mising_values

```
std::unordered_map<int, int> DF::DataFrame::mising_values
```

an unorderd maps for missing values

Definition at line 74 of file ReadFiles.hpp.

5.1.3.4 n_cols

```
unsigned long long DF::DataFrame::n_cols
```

number of columns

Definition at line 68 of file ReadFiles.hpp.

5.1.3.5 n_rows

```
unsigned long long DF::DataFrame::n_rows
```

number of rows

Definition at line 62 of file ReadFiles.hpp.

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

- include/ReadFiles.hpp
- src/ReadFiles.cpp

Chapter 6

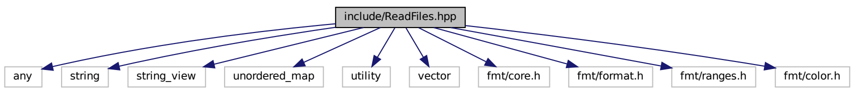
File Documentation

6.1 include/ReadFiles.hpp File Reference

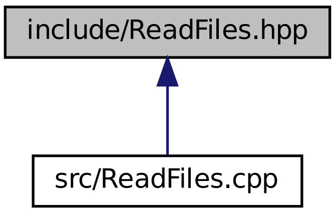
A class for reading files with different delimiters.

```
#include <any>
#include <string>
#include <string_view>
#include <unordered_map>
#include <utility>
#include <vector>
#include "fmt/core.h"
#include "fmt/format.h"
#include "fmt/ranges.h"
#include "fmt/color.h"
```

Include dependency graph for ReadFiles.hpp:



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



Classes

- class DF::DataFrame

  DataFrame is class for parsing data in a given file with a give delimeter (default is comma ','). all data will be saved as string wich user can later covert to desired type.

Namespaces

- std
- std::shorts

  namespace for introducting shortnames
- DF

Typedefs

- using std::shorts::V_string = vector< string >
- using std::shorts::VV_string = vector< V_string >
- using std::shorts::V_any = vector< any >
- using std::shorts::VV_any = vector< vector< any > >
- using std::shorts::Data = unordered_map< string, V_string >
- using std::shorts::V_double = vector< double >
- using std::shorts::V_int = vector< int >
- using std::shorts::V_pair_ints = std::vector< std::pair< int, int > >

6.1.1 Detailed Description

A class for reading files with different delimiters.

Author

Naeim Moafinejad (`snmoafinejad@iimcb.gov.pl`, `s.naeim.moafi.n@gmail.com`)

Version

0.1

Date

2024-10-29

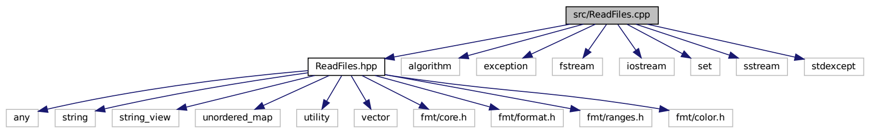
Copyright

Copyright (c) 2024

6.2 src/ReadFiles.cpp File Reference

```
#include "ReadFiles.hpp"
#include <algorithm>
#include <exception>
#include <fstream>
#include <iostream>
#include <set>
#include <sstream>
#include <stdexcept>
```

Include dependency graph for ReadFiles.cpp:



Functions

- int main ()

6.2.1 Function Documentation

6.2.1.1 main()

```
int main ( )
```

Definition at line 327 of file ReadFiles.cpp.

```
{
    std::string software_name = R"( ____        _        _____
|  _ \  __ _| |_ __ _|  ___| __ __ _ _ __ ___   ___
| | | |/ _` | __/ _` | |_ | '__/ _` | '_ ` _ \ / _ \
| |_| | (_| | || (_| |  _|| | | (_| | | | | | |  __/
|____/ \__,_|\__\__,_|_|  |_|  \__,_|_| |_| |_|\___|)";

        std::cout « software_name « std::endl « std::endl;
    DF::DataFrame df;


    df.read_files("test.txt", '\t');
    df.head();
    std::cout « '\n';

    auto df2 = df.copy_by_headers({"id", "age", "disease"});
    df2.head();
    std::cout « '\n';

    df2.swap_cols_pos("age", "disease");
    df2.head();
    std::cout « '\n';

    df.read_files("test.txt", '\t', false);
    df.head();
    std::cout « '\n';

    df.read_files("test.txt", '\t', false, {"1.0000", "2.0000", "3.0000", "4.0000", "5.0000"});
    df.head();
    std::cout « '\n';

    df.head(600);
    std::cout « '\n';

    DF::DataFrame df3;
    std::vector<std::string> v_hdrs {"group_PDB", "id", "label_atom_id", "label_comp_id", "label_asym_id",
                                     "label_seq_id", "Cartn_x", "Cartn_y", "Cartn_z", "occupancy",
                                     "B_iso_or_equiv", "type_symbol", "charge"};

    std::shorts::V_pair_ints fields_intervals = {{0,6}, {6,5}, {12,4}, {17,3}, {20,2}, {22,4}, {30,8}, {38,8}, {46,8}, {54,6}, {60,6}, {76,2}, {78,2}};

    // std::string text{"HETATM    1  PG  GTP A   1      24.181  32.064  27.670  0.10 24.73           P \nHETATM    2  O1G GTP A   1      24.342  33.433  27.064  0.10 37.56           O  \nHETATM    3  O2G GTP A   1      24.519  32.013  29.136  0.10 32.46           O  \n"};
    std::string text = R"(HETATM    1  PG  GTP A   1      24.181  32.064  27.670  0.10 24.73           P

HETATM    2  O1G GTP A   1      24.342  33.433  27.064  0.10 37.56           O
HETATM    3  O2G GTP A   1      24.519  32.013  29.136  0.10 32.46           O
HETATM    4  O3G GTP A   1      25.048  31.062  26.907  1.00 47.91           O
HETATM    5  O3B GTP A   1      22.644  31.665  27.526  1.00 33.11           O )";

    df3.read_text(text, fields_intervals, false, v_hdrs);
    df3.head();


    return EXIT_SUCCESS;
}
```

Index