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 Data = unordered_map< string, V_string >
- using V_double = vector< double >
- using V_int = vector< 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 35 of file ReadFiles.hpp.

4.3.2.2 V_double

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

Definition at line 36 of file ReadFiles.hpp.

4.3.2.3 V_int

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

Definition at line 37 of file ReadFiles.hpp.

4.3.2.4 V_string

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

Definition at line 33 of file ReadFiles.hpp.

4.3.2.5 VV_string

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

Definition at line 34 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_files (std::string_view path, std::shorts::V_int const &v_cols_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)
- void fill_data (std::shorts::V_string const &v_strs, char delim=',', bool is_first_col_header=true, std::shorts::V_string v_hdrs={})
- void fill_data (std::shorts::V_string const &v_strs, std::shorts::V_int const &v_cols_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 49 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 173 of file ReadFiles.cpp.

```
174 {
175     DF::DataFrame new_df;
176     new_df.n_cols = v_hdrs.size();
177     new_df.n_rows = n_rows;
178     new_df.headers = v_hdrs;
179 
180     for(auto const& hdr : v_hdrs)
181     {
182         new_df.data[hdr] = data[hdr];
183     }
184 
185     return new_df;
186 }
```

5.1.2.3 fill_data() [1/2]

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

Definition at line 48 of file ReadFiles.cpp.

```
49 {
50     std::shorts::VV_string vv_strs;
51 
52     for(auto const& line : lines)
53     {
54         std::istringstream iss(line);
55         std::string cell;
56         std::shorts::V_string v_str_tmp;
57 
58         while(std::getline(iss, cell, delim))
59         {
60             cell.erase(std::remove(cell.begin(), cell.end(), '\"'), cell.end());
61             v_str_tmp.emplace_back(cell);
62         }
63         vv_strs.emplace_back(v_str_tmp);
64     }
65 
66     // init n_rows and n_cols
67     n_rows = vv_strs.size();
68     n_cols = vv_strs[0].size();
69 
70     headers.resize(n_cols);
71 
72     // initializing headers
73     for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
74     {
75         if(is_first_col_header)
76         {
77             headers[i_col] = vv_strs[0][i_col];
78         }
79         else if(v_hdrs.size() > 0)
80         {
81             if(v_hdrs.size() == n_cols)
82             {
83                 headers[i_col] = v_hdrs[i_col];
84             }
85             else
86             {
```

```
87                 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"));
88             }
89         }
90         else
91         {
92             headers[i_col] = std::to_string(i_col + 1);
93         }
94     }
95
96     for(unsigned long long i_col{0}; i_col < n_cols; ++i_col)
97     {
98         std::shorts::V_string values;
99         for(unsigned long long i_row{is_first_col_header}; i_row < n_rows; ++i_row)
100         {
101             if(vv_strs[i_row].size() != n_cols)
102             {
103                 throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: inconsistent number of
       columns, check row {}", i_row + 1));
104                 exit(EXIT_FAILURE);
105             }
106
107             values.emplace_back(vv_strs[i_row][i_col]);
108
109             // check for the missig values
110             // missing values are empty string, NA, and NAN
111             if(vv_strs[i_row][i_col] == "" || vv_strs[i_row][i_col] == "NA" || vv_strs[i_row][i_col] ==
       "NAN")
112             {
113                 mising_values.insert({i_row, i_col});
114             }
115         }
116
117         data[headers[i_col]] = values;
118     }
119 }
```

5.1.2.4 fill_data() [2/2]

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

5.1.2.5 get_by_header()

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

Definition at line 168 of file ReadFiles.cpp.

```
169 {
170     return data[hdr];
171 }
```

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 163 of file ReadFiles.cpp.

```
164 {
165     return headers;
166 }
```

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.

```
13 {
14     return n_cols;
15 }
```

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.

```
18 {
19     return n_rows;
20 }
```

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 127 of file ReadFiles.cpp.

```
128 {
129     if(n > n_rows)
130     {
131         fmt::print(fg(fmt::color::yellow),"Warning: number of columns requested ({}) to be printed is
      bigger than number of available rows in data ({})\n",
132         n, data[headers[0]].size());
133         if(n_rows > 5) n = 5;
134         else n = n_rows;
135     }
136 
137     for(auto const& curr_hdr : headers)
138     {
139         fmt::print(fmt::emphasis::bold | fg(fmt::color::green), "{:10} ", curr_hdr);
140     }
141 
142     std::cout « '\n';
143 
144     for(unsigned long long i{0}; i < n; ++i)
145     {
146         for(auto const& curr_hdr : headers)
147         {
148             if(data[curr_hdr][i] == "" || data[curr_hdr][i] == "NA" || data[curr_hdr][i] == "NAN")
149             {
150                 fmt::print(bg(fmt::color::red),"{:10} ", data[curr_hdr][i]);
151             }
152             else
153             {
154                 fmt::print("{:10} ", data[curr_hdr][i]);
155             }
156 
157         }
158 
159         std::cout « "\n";
160     }
161 }
```

5.1.2.10 read_files() [1/2]

```
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 121 of file ReadFiles.cpp.

```
122 {
123     auto v_strs = read_lines(path);
124     fill_data(v_strs, delim, is_first_col_header, v_hdrs);
125 }
```

5.1.2.11 read_files() [2/2]

```
void DF::DataFrame::read_files (
            std::string_view path,
            std::shorts::V_int const & v_cols_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	

5.1.2.12 read_lines()

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

Definition at line 22 of file ReadFiles.cpp.

```
23 {
24     std::ifstream ifs(path.data());
25 
26     if(ifs.fail())
27     {
28         throw std::runtime_error(fmt::format(fg(fmt::color::red), "Error: unable to read file {}.\nPlease
       check your input.", path));
29         exit(EXIT_FAILURE);
30     }
31 
32 
33     std::string line;
34     std::shorts::V_string v_strs;
35 
36 
37     while(std::getline(ifs, line))
38     {
39         if(line.size() == 0) continue;
40         line.erase(std::remove(line.begin(), line.end(), '\r'),line.end());
41         v_strs.emplace_back(line);
42     }
43 
44     return v_strs;
45 }
```

5.1.2.13 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

5.1.2.14 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

5.1.2.15 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 188 of file ReadFiles.cpp.

```
189 {
190     std::string tmp_hdr;
191     std::shorts::V_string tmp_vals;
192     auto first_it = std::find(headers.begin(), headers.end(), first_hdr);
193     auto second_it = std::find(headers.begin(), headers.end(), second_hdr);
194     swap(headers[first_it - headers.begin()], headers[second_it - headers.begin()]);
195
196     std::swap(data[first_hdr], data[second_hdr]);
197 }
```

5.1.3 Member Data Documentation

5.1.3.1 data

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

Definition at line 158 of file ReadFiles.hpp.

5.1.3.2 headers

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

Definition at line 159 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 69 of file ReadFiles.hpp.

5.1.3.4 n_cols

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

number of columns

Definition at line 63 of file ReadFiles.hpp.

5.1.3.5 n_rows

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

number of rows

Definition at line 57 of file ReadFiles.hpp.

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

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

Chapter 6

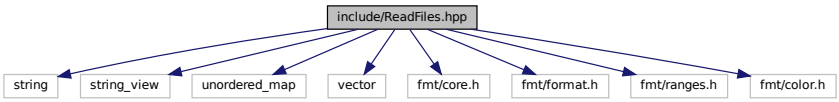
File Documentation

6.1 include/ReadFiles.hpp File Reference

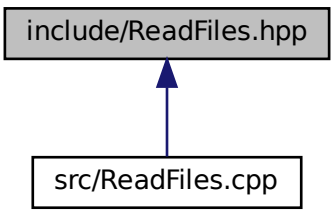
A class for reading files with different delimiters.

```
#include <string>
#include <string_view>
#include <unordered_map>
#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::Data = unordered_map< string, V_string >
- using std::shorts::V_double = vector< double >
- using std::shorts::V_int = vector< 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

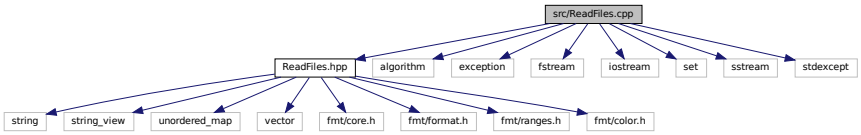
2024-10-29

Copyright

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 201 of file ReadFiles.cpp.

```
202 {
203     DF::DataFrame df;
204 
205 
206     df.read_files("test.txt", '\t');
207     df.head();
208     std::cout << '\n';
209 
210     auto df2 = df.copy_by_headers({"id", "age", "disease"});
211     df2.head();
212     std::cout << '\n';
213 
214     df2.swap_cols_pos("age", "disease");
215     df2.head();
216     std::cout << '\n';
217 
218     df.read_files("test.txt", '\t', false);
219     df.head();
220     std::cout << '\n';
221 
222     df.read_files("test.txt", '\t', false, {"1.0000", "2.0000", "3.0000", "4.0000", "5.0000"});
223     df.head();
224     std::cout << '\n';
225 
226     df.head(600);
227 
228 
229     return EXIT_SUCCESS;
230 }
```

Index