CUDDLE Library Documentation

CUDDLE - A C Implementation of Pandas-like Data Manipulation

Table of Contents

- Overwiew
- Getting Started
- Core Functionality
- Basic Data Operations
- API Reference

Overview

CUDDLE is a static library (libcuddle.a) that provides robust data manipulation capabilities in C. It's designed to handle CSV files and perform various operations similar to Python's pandas library.

Key Features

- CSV file reading and writing with custom separators
- Multiple data type support (bool, int, uint, float, string)
- Data filtering and sorting
- Aggregation operations
- Data transformation capabilities

Getting Started

Building the Library

```
make        # Builds the library
make re     # Rebuilds the library
make clean  # Removes object files
make fclean # Removes object files and binary
make test, cov, tests_run...
```

Basic Usage

```
#include "dataframe.h"

int main(void) {
    dataframe_t *df = df_read_csv("data.csv", NULL);
    // Perform operations
    df_free(df);
    return 0;
}
```

Core Functionality

Data Types

```
typedef enum {
    BOOL,
    INT,
    UINT,
    FLOAT,
    STRING,
    UNDEFINED
} column_type_t;
```

Data Structure

```
typedef struct dataframe_s {
    int nb_rows;
    int nb_columns;
```

```
    // Internal implementation details
} dataframe_t;
```

Data Operations

Basic Operations

Operation	Function	Description
Read CSV	df_read_csv()	Reads data from a CSV file
Write CSV	df_write_csv()	Writes data to a CSV file
Head	df_head()	Returns first n rows
Tail	df_tail()	Returns last n rows
Shape	df_shape()	Returns dimensions of the dataframe

API Reference

Core Functions

▼ **df_read_csv()**

Reads a CSV file into a dataframe structure.

```
dataframe_t *df_read_csv(const char *filename, const char *separator);
```

- Parameters:
  - filename: Path to the CSV file
  - separator: Custom separator (NULL for comma)
- Returns: Pointer to new dataframe or NULL on error

▼ **df_write_csv()**

Writes a dataframe to a CSV file.

```
int df_write_csv(dataframe_t *dataframe, const char *filename);
```

- Parameters:
  - dataframe: Pointer to dataframe
  - filename: Output file path

- Returns: 0 on success, non-zero on error

Data Manipulation

▼ **df_filter()**

Filters rows based on a condition.

```
dataframe_t *df_filter(dataframe_t *dataframe, const char *column, bool (*
```

Example usage:

```
bool filter_age_above_30(void *value) {
    return *(int*)value > 30;
}

dataframe_t *filtered = df_filter(df, "age", filter_age_above_30);
```

▼ **df_sort()**

Sorts the dataframe based on a column.

```
dataframe_t *df_sort(dataframe_t *dataframe, const char *column, bool (*
```

▼ **df_to_type()**

Change and parse type based on a column.

```
dataframe_t *df_to_type(dataframe_t *df, char const *column, column_typ
```

Error Handling

All functions return NULL or error code 84 in case of failure. Error messages are written to stderr.

Best Practices

- Always free dataframes using df_free() when no longer needed
- Check return values for NULL to handle errors