Online Data Analysis Documentation

Release 1.0.0

Alastair Chin

October 16, 2015

THE UNIVERSITY OF WESTERN AUSTRALIA

Achieve International Excellence

This contains the documentation for the online data analysis project

CHAPTER

ONE

REQUIREMENTS

This requires Python 3, scipy, numpy and pandas which can be installed by using the Anaconda package which contains all necessary files.

The program currently consists of 6 files

Contents:

1.1 Application

Main execution body for program.

`application.`**`get_file_dir`**`(`*location*`)`
Returns the directory of the file with the file name

Keyword arguments: location – A file path.

`application.`**`main`**`(`**args*`)`
Create Data and Report objects, providing necessary information for them to run analysis and create desired outputs (i.e. HTML report).

Keyword Arguments: args – Arguments provided to the program at runtime.

1.2 Data

Reads CSV file for information, provides basic cleaning of data and then runs analysis on said data.

Global Variables: threshold – A float representing a percentage of which a columns values must be of a type before the column is declared to be of that type i.e. if 95% of the columns values are integers it will be of type Integer. Can be set via template, default 0.9 (90%).

enum_threshold – An integer representing the amount of times which a value in an Enumerable type column must appear for it to not be taken as an error. Can be set via template, default 1.

invalid_values – Values which are pre-agreed not to appear in valid data and should be stripped. Currently not used.

re_float – Regular expression for float type.

re_int – Regular expression for integer type.

re_email – Regular expression for email type.

re_currency – Regular expression for currency type.

re_boolean – Regular expression for boolean type.

re_sci_notation – Regular expression for scientific notation type.

re_separation – Regular expression for the different types of delimiters in files.

re_date – Regular expression for date type.

re_time – Regular expression for time type.

re_char – Regular expression for character type.

re_day – Regular expression for day type.

re_hyper – Regular expression for hyperlink type.

class data.**Column**(*header=''*)

Object to hold data from each column within the provided CSV file.

Methods: change_misc_values – Removes misc/unclear values from column values. Used in Data.clean() but this function is unused.

drop_greater_than – Removes '<', '>' from column values. Defined but unused.

define_most_least_common – Sets object variable to hold 15 most common values and least common values for that column.

define_type – Sets object variable to type (e.g., String) according to column values.

define_errors – Defines a list that contains the row and column of possibly incorrect values.

check_empty – Checks whether a provided cell in a column is empty or not.

set_type – Sets type of column for use with templates

set_size – Sets the size of the data for use when checking for errors, for use with the 'Identifier' data type.

set_empty – Sets a columns empty attribute to True.

set_not_empty – Sets a columns empty attribute to False.

is_Empty – Returns whether or not a columns empty attribute is True or False.

set_Identifier_size – Sets the size of the data for identifier type.

updateCell – Changes the value of a given cell with one provided.

Variables: most_common – List with the <= 15 most common results within the column values.

least_common – List with the <= 15 least common results within the column values.

empty – Boolean value of whether the column holds values or not.

header – String containing column header/title.

type – The type of data in column, e.g., String, Float, Integer, Enumerated, represented as a String.

values – List of CSV values for the column.

analysis – Analysis object associated with this column.

unique – Integer representing the amount of unique values in this column.

total_true – The amount of 'true' booleans in this column.

total_false – The amount of 'false' booleans in this column.

total_yes – The amount of 'yes' booleans in this column.

total_no – The amount of 'no' booleans in this column.

data_size – The length which all correct strings in this column should be.

ignore_empty – A boolean representing whether empty cells in this column should be ignored.

change_misc_values()
Replaces identified values of unclear meaning or inexact value, i.e., '-', with an agreed value.

check_empty(*x*, *value*, *columnNumber*, *errors*, *formatted_errors*, *invalid_rows_pos*, *set_to_ignore*, *data_start*)
Checks an individual cell of a column to see if it is empty. If it is set to ignore empty cells for this column returns True. If it is not set to ignore, return True and add cell to the list of errors. If it is not empty, return False.

Keyword arguments: x – Integer postion of cell in column.

value – value in cell.

columnNumber – Number of column that cell is in.

errors – List of errors. Numbered from 0. Contains row number (formatted for data with incorrect columns removed), column number, cell value, reason for error (empty cell) and row number (formatted for data with incorrect columns still present).

formatted_errors – List of formatted errors, a human readable version of errors. Contains a string listing the row and column in human readable form of the empty cell and it's reason for being an error (empty cell).

invalid_rows_pos – An array containing a number matching the amount of invalid rows that have been removed from analysis by the time that row is accessed. i.e. invalid_rows_pos[1] = 2 says that by the time values[1] is evaluated two rows have been removed from analysis.

set_to_ignore – A set containing integers representing columns set to ignore empty values in.

data_start – Integer representing the row actual data (not headers) starts on.

define_errors(*columnNumber*, *errors*, *formatted_errors*, *invalid_rows_pos*, *range_list2*, *set_to_ignore*, *data_start*)
Define all the rows/columns with invalid values and append to errors, and formatted_errors once formatted properly.

Keyword arguments: columnNumber – The number of the current column being iterated over, numbered from 0.

errors – A list of errors to be edited of the form (row number, column number, error value) which is numbered from 0.

formatted_errors – A list of errors to be edited of the form (row number, column number, error value) which is numbered from 1.

invalid_rows_pos – An array containing a number matching the amount of invalid rows that have been removed from analysis by the time that row is accessed. i.e. invalid_rows_pos[1] = 2 says that by the time values[1] is evaluated two rows have been removed from analysis.

range_list2 – A list with two values (min, max) respectively, if supplied in a template all values numeric values must fall between these two values or are an error.

set_to_ignore – A set containing integers representing columns set to ignore empty values in.

data_start – Integer representing the row actual data (not headers) starts on.

define_most_least_common()
Set 15 most common results to class variable, and set object variable empty if appropriate.

define_type()
Run column data against regex filters and assign object variable type as appropriate.

is_Empty()
Whether or not column is empty

set_Identifier_size(*size*)
Sets the size of the data for identifier type

set_empty()
Set Column to be empty

set_not_empty()
Set Column to be not empty

set_size(*size*)
Sets the size of the data for use when checking for errors. For use with the 'Identifier' data type

size – length of identifier

set_type(*type*)
Sets type of column for use with templates

uniqueCount(*values*)
Return the amount of unique values in the values list.

Keyword arguments: values – A list of values.

updateCell(*pos*, *new_value*)
Changes the value of a cell given

Keyword Arguments:

pos – position of cell in column to change

new_value – value to set cell too

class data.**Data**(**args*)
Main store for CSV data, reading the data from the CSV file and then assigning out to relevant variables.

Methods: read – Reads the CSV file and outputs to raw_data variable.

remove_invalid – Reads from raw_data variable and assigns rows to valid_rows or invalid_rows according to their length.

create_columns – Creates column object according to valid_rows, assigning column header and column values.

clean – Calls column cleaning methods to run 'cleaning' on all columns.

analysis – Calls column analysis methods to run 'analysis' on all columns.

find_errors – Iterates through all columns in the Data object and calls these columns define_errors function.

pre_analysis – Iterates through a Data objects columns and first defines their least and most common elements, then if template is supplied, sets the type of the column to match the template, if not if column is not empty defines its type, and if it's a Identifier data type sets the columns size to me no more than data_size.

gen_file – Generates a csv file based on the data for after data has been corrected.

get_row – Returns the value of a given row in a list.

change_row – Edits a row of the data to a given value.

getCellErrors – Returns list of all cells containing invalid data, contains row number,. column number and its value.

getRowErrors – Returns a list of all row errors

getColumns – Returns a list of all columns

get_column – Returns a column of the Data given a column number.

get_headers – Returns the header row of the data.

set_headers – Given a map of column numbers to header names, maps the column headers to the correct value.

clear_errors – Clears Data.errors and Data.formatted_errors to allow find_errors() to be rerun.

rebuild_raw_data – Recreates raw_data from Data's columns and row.

delete_invalid_row – Deletes given invalid row at the index from the data.

Variables:

analysers – Dictionary conataining types as keys and their respective analysers as values (i.e. analysers['type'] == TypeAnalyser

types – Tuple containing all valid types as ordered pairs of form ('Type', 'Human readable type'). Used to map types on web site to their correct progammatic name.

Filename – String of path to file containing data

columns – List of column objects.

invalid_rows – List of invalid rows (i.e., more or less columns than number of headers). Copied from raw_data

invalid_rows_indexes – List of indexes corresponding to invalid rows.

formatted_invalid_rows – List of invalid rows for report.

invalid_rows_pos – List of the amount of invalid rows in the raw data prior to each valid row (i.e. the nth element contains number of invalid rows prior to the nth valid row)

errors – list of errors in file; errors[n][0] is row of error, errors[n][1] is column of error, errors[n][2] is the value of in that location, errors[n][3] is the reason for the error & error[4] is the index for the value in columns[n1].values.

formatted_errors – List of errors in file, each error contains: row, column and value of the error.

raw_data – List of raw CSV data as rows. After remove_invalid() has run this only contains rows from the CSV file prior to the start of the data.

valid_rows – List of valid rows (i.e., same number of columns as headers).

can_edit_rows – is a boolean value true after remove_invalid() and before create_columns() have been run only. It defines whether rebuild_raw_data() and delete_invalid_row() may be called.

data_in_columns – is a boolean true after create_columns() is completed, it defines whether the Data object is in a form gen_file() expects.

datatypes_are_defined – is a boolean true after pre_analysis() has been run and each column's type is defined. It defines whether export_datatypes() may be run.

template – The template containing various settings.

delimiter_type – The delimiter used in the csv file (space, comma, tab, colon etc)

header_row – The row the headers (non-used data) is on.

self.data_start – The row the used data starts on.

data_size – The length all Identifier types Strings should be.

ignore_empty – A boolean stating whether empty columns should be skipped.

std_devs_val – The amount of standard deviations from the mean a value should be before it is counted as an error i.e. (mean +- std_devs_val * std_dev)

range_list – A list representing the minimum and maximum allowed values for any numeric data, outside of which it is an error. Formatted (min, max).

set_ignore – A set of integers representing columns to ignore empty values in.

analysis()
Iterates through each column and analyses the columns values using the columns type analyser.

change_row(*row_num*, *new_values*)

Edits a row of the data to a given value. Keyword Arguments:

row_num - number of row being changed

new_values - list of values row is to be changed to

clean()
Calls cleaning methods on all columns.

clear_errors()
Wipes recorded errors to allow find_errors() to be rerun

create_columns()
For each row in raw_data variable, assigns the first value to the headers variable and creates a Column object with that header provided. Then removes header row from valid_rows. Then for each row in valid_rows, populates relevant column object with row data.

delete_invalid_row(*invalid_row_index*)
Deletes given invalid row at the index from the data.

Keyword Arguments:

invalid_row_index – Index of invalid row to be deleted.

find_errors()
Iterates through each column and finds any errors according to pre-determined conditions.

gen_file(*filePath=''*)
Generates a csv file based on the data for after data has been corrected

Keyword Arguments:

filePath – Name of file to be generated.

getCellErrors()
Returns list of all cells containing invalid data, contains row number,. column number and its value.

getColumns()
Returns a list of all columns

getRowErrors()
Returns a list of all row errors

get_column(*colNo*)
Returns a column of the data given a column number

get_headers()
Returns the headers of data

get_row(*row_num*)
Returns the values of a row in list

Keyword Arguments:

row_num – The row number to be fetched

pre_analysis()
First defines their least and most common elements, then if template is supplied, sets the type of the column to match the template, if not if column is not empty defines its type, and if it's a special data type sets the columns size to me no more than data_size.

read(*csv_file*)
Opens and reads the CSV file, line by line, to raw_data variable.

Keyword arguments: csv_file – The filename of the file to be opened.

rebuild_raw_data()
Re creates raw_data from Data's columns and row.

remove_invalid()
For each row in raw_data variable, checks row length and appends to valid_rows variable if same length as headers, else appends to invalid_rows variable. invalid_rows_indexes holds the amount of rows that have been skipped by the point the xth row has been accessed from valid_rows.

set_headers(*header_map*)
Sets headers of columns taking a dictionary mapping column numbers to headers.

Keyword Arguments: header_map – A map of column numbers to headers.

1.3 Report

Generate reports based on data provided via a Data object.

Classes: Report – Contains methods to generate and output appropriate HTML for the report.

class report.**Report**(*data*, *file*)
The main report object.

Methods: __init__ – Initialise the object and create required local variables.

empty_columns – Return empty columns in the data object.

html_report – Create HTML report and output to file.

list_creator – Helper method to generate a HTML list from provided input.

row_creator – Helper method to generate HTML rows from provided input.

numerical_analysis – Return numerical based statistics on input.

string_analysis – Return string based statistics on input.

enum_analysis – Return enumeration based statistics on input.

bool_analysis – Return boolean based statistics on input.

email_analysis – Return email based statistics on input.

date_analysis – Return date based statistics on input.

time_anaysis – Return time based statistics on input.

day_analysis – Return day based statistics on input.

hyper_analysis – Return hyperlink based analyis on input.

list_creator – Provided a list, returns an unordered html list of values in the list.

Static Methods: list_creator – Provided a list, returns an unordered html list of values in the list.

row_creator – Provided a list, returns a HTML row of values in the list.

initial_show_items – Returns the number of items to show initially for each type in the report before hiding them under a 'show more' button.

Variables: GRAPH_LIMIT – How many rows before the graphs will stop displaying every value, but just show a summary

data – Reference to Data object, the output from analysis of file_name.

file_name – Reference to a CSV file containing the data being worked on.

chart_data – A String of data that is formatted correctly to be input to a graph API.

boolean_analysis()
Return HTML string of boolean analysis on columns of type boolean in the data objectby accessing the various class variables of the columns.

char_analysis()
Return HTML string of char analysis on columns of type char in the data object by accessing the various class variables of the columns.

currency_analysis()
Return HTML string of numerical analysis on columns of type Currency in the data object by accessing the various class variables of the columns.

date_analysis()
Return HTML string of date analysis on columns of type date in the data object by accessing the various class variables of the columns.

day_analysis()
Return HTML string of day analysis on columns of type day in the data object by accessing the various class variables of the columns.

email_analysis()
Return HTML string of email analysis on columns of type email in the data objectby accessing the various class variables of the columns.

empty_columns()
Return a list of empty columns in the data object.

enum_analysis()
Return HTML string of enumeration analysis on columns of type Enum in the data object by accessing the various class variables of the columns.

gen_html(*html*)
Generates html report for the file

html_report()
Write a HTML file based on analysis of CSV file by calling the various type analyses. Returns a string of html.

hyper_analysis()
Return HTML string of hyperlink analysis on columns of type hyper in the data object by accessing the various class variables of the columns.

identifier_analysis()
Return HTML string of identifier analysis on columns of type identifier in the data objectby accessing the various class variables of the columns by accessing the various class variables of the columns.

static initial_show_items()
Return the number of items to show initially, where clicking 'show more' will expand.

static list_creator(*list_items*)
Return provided list as an unordered HTML list.

Keyword arguments: list_items – List of items to be turned into HTML.

numerical_analysis()
Return HTML string of numerical analysis on columns of type Float or Integer in the data object by accessing the various class variables of the columns.

static row_creator(*row_items*, *rowNumber=0*, *type='none'*)
Return provided list as HTML rows.

Arguments: row_items – List of items to be turned into HTML.

string_analysis()
Return HTML string of string analysis on columns of type string in the data object by accessing the various class variables of the columns.

time_analysis()
Return HTML string of time analysis on columns of type time in the data object by accessing the various class variables of the columns.

1.4 Template

Provide a base HTML template variable for population with appropriate statistics in the report.py module.

1.5 Analyser

Analyser class for running analysis on columns depending on the column type

class analyser.**Analyser**(*values*)
Base analysis class object. Initiate the object, and assigns the statistical mode, if any.

Global variables: max_Outliers – the maximum amount of outliers that will be found.

standardDeviations – The number of standard deviations away from the mean a value is allowed to be before it is an error, default 3.

re_date – A regular expression for dates.

re_dateDF – A regular expression for months December-February.

re_dateMM – A regular expression for months March-May

re_dateJA – A regular expression for months June-August.

re_dateSN – A regular expression for months September-November.

re_time – A regular expression for time.

re_timePM – A regular expression for PM times.

re_timeAM – A regular expression for AM times

re_timehr – A regular expression for the hour.

Class variables: mode – Returns the mode of the column analysed.

unique – The count of unique values in the column.

Child classes and associated variables: StringAnalyser – String column analysis.

EmailAnalyser – Email column analysis.

EnumAnalyser – Enumerated column analysis.

NumericalAnalyser – String/Float column analysis. min – Minimum value in column values.

max – Maximum value in column values.

mean – Mean value in column values.

median_low – Low median for column values.

median – Median value for column values.

median_high – High median for column values.

normDist – String Yes/No if columns value is normally distributed.

stdev – Standard deviation for column values, N/A if not normally distributed to within 95.5% confidence.

stDevOutliers – List of values outside a certain number of standard deviations from the mean.

CurrencyAnalyser – Child class of NumericalAnalyser

BooleanAnalyser – Boolean column analysis

DateAnalyser – Date column analysis

TimeAnalyser – Time column analysis

CharAnalyser – Character column Analysis

DayAnalyser – Day column Analysis

HyperAnalyser – Hyperlink column Analysis

Class Methods: uniqueCount – Returns the count of unique values in a list.

uniqueCount(*values*)
Return the amount of unique values in the values list.

Keyword arguments: values – A list of values.

class analyser.**BooleanAnalyser**(*values*)
Run boolean analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

class analyser.**CharAnalyser**(*values*)
Run char analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

class analyser.**CurrencyAnalyser**(*values*, *stdDevs*)
Run currency analysis, using NumericalAnalyser as a super class. Removes currency symbols in values.

Keyword arguments: NumericalAnalyser – A NumericalAnalyser object.

class analyser.**DateAnalyser**(*values*)
Run date analysis, currently only using Analyser super class methods.

Keyword Arguments: Analyser – An analyser object.

class analyser.**DayAnalyser**(*values*)
Run day analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

class analyser.**EmailAnalyser**(*values*)
Run email analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

class analyser.**EnumAnalyser**(*values*)
Run enumerated analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

class analyser.**HyperAnalyser**(*values*)
Run hyperlink analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

class analyser.**IdentifierAnalyser**(*values*)
Run identifier analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

class analyser.**NumericalAnalyser**(*values*, *stdDevs*)
Runs numeric analysis.

Keyword arguments: Analyser – An analyser object.

class analyser.**SciNotationAnalyser**(*values*, *stdDevs*)
Run scientific notation analysis.

Keyword arguments: Analyser – An analyser object.

Class Methods: int_to_sci – Converts a a given number into a string in scientific notation form.

int_to_sci(*value*)
Converts numbers into a string in scientific notation form

Keyword arguments: value – The value to be converted to scientific notation.

class analyser.**StringAnalyser**(*values*)
Run string analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

class analyser.**TimeAnalyser**(*values*)
Run time analysis, currently only using Analyser super class methods.

Keyword arguments: Analyser – An analyser object.

1.6 Template Reader

Class for reading templates to pass on information about how to process the data for the data class

class template_reader.**Template**(*filename*)
Object storing user input that describes data given. Able to specify:

Class Variables: columns – state column number and data type.

delimiter_type – state delimiter character (for comma and space use the word not ',' or ' ').

header_row – row of header (0 for no header).

data_start – row that data starts on

data_size – Length that all strings of identifer type must be.

ignore_empty – Boolean representing whether to ignore empty cells or not.

threshold_val – minimum proportion of column that has the correct data type

enum_threshold_val – Minimum amount of times an enumerated value must appear in a column to not be an error.

std_devs – How many standard deviations away from the mean numeric values are allowed to be.

range_vals – A list of two items [min, max] representing the minimum and maximum values numeric values can take.

ignore_set – A set listing all the columns that empty cells are to be ignored in.

Columns and rows start at 1 not 0

Class Methods: read – Reads the template csv file and inputs data into corresponding variables if it's found in the file.

read (*filename*)

Reads template file, assumes correct formatting, if user editing is permitted will need to be improved with more checks.

Keyword Arguments: filename – filename of csv template file to be read for options.

See documentation

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

a

d

r

t

S

T

U