IQDM-Analytics

Release 0.1.9

Dan Cutright

May 17, 2021

CONTENTS

CHAPTER

ONE

IQDM-ANALYTICS

1.1 What does it do?

IQDM Analytics is a desktop application that mines IMRT QA reports with IQDM-PDF and performs statistical analysis.

1.2 Executables

Single-file executables are available. See attachments in the latest release.

1.3 Other information

This library is part of the IMRT QA Data Mining (IQDM) project for the AAPM IMRT Working Group (WGIMRT).

- Free software: MIT license
- Documentation: Read the docs

1.4 Dependencies

- iqdmpdf - Mine IMRT QA PDF's
- wxPython Phoenix - Build a native GUI on Windows, Mac, or Unix systems
- Bokeh - Interactive Web Plotting for Python
- NumPy - The fundamental package for scientific computing with Python
- selenium - A browser automation framework and ecosystem
- PhantomJS - PhantomJS is a headless web browser scriptable with JavaScript
- pypubsub - A Python publish-subscribe library

1.5 Install and Run

If you prefer to run from source:

```
$ git clone https://github.com/IQDM/IQDM-Analytics.git
$ cd IQDM-Analytics
$ python iqdma_app.py
```

Note you *may* have to use pythonw instead of python, depending on your version.

1.6 TODO

- Ability to cancel PDF-Miner thread
- Unit testing (non-GUI)
- Setup continuous integration
- Consolidate/Clean-up UserSettings Dialog code

1.7 Credits

1.7.1 Development Lead

- Dan Cutright, University of Chicago Medicine

1.7.2 Contributors

- Marc Chamberland, University of Vermont Health Network
- Serpil Kucuker Dogan, Northwestern Medicine
- Mahesh Gopalakrishnan, Northwestern Medicine
- Aditya Panchal, AMITA Health
- Michael Snyder, Beaumont Health

CHAPTER

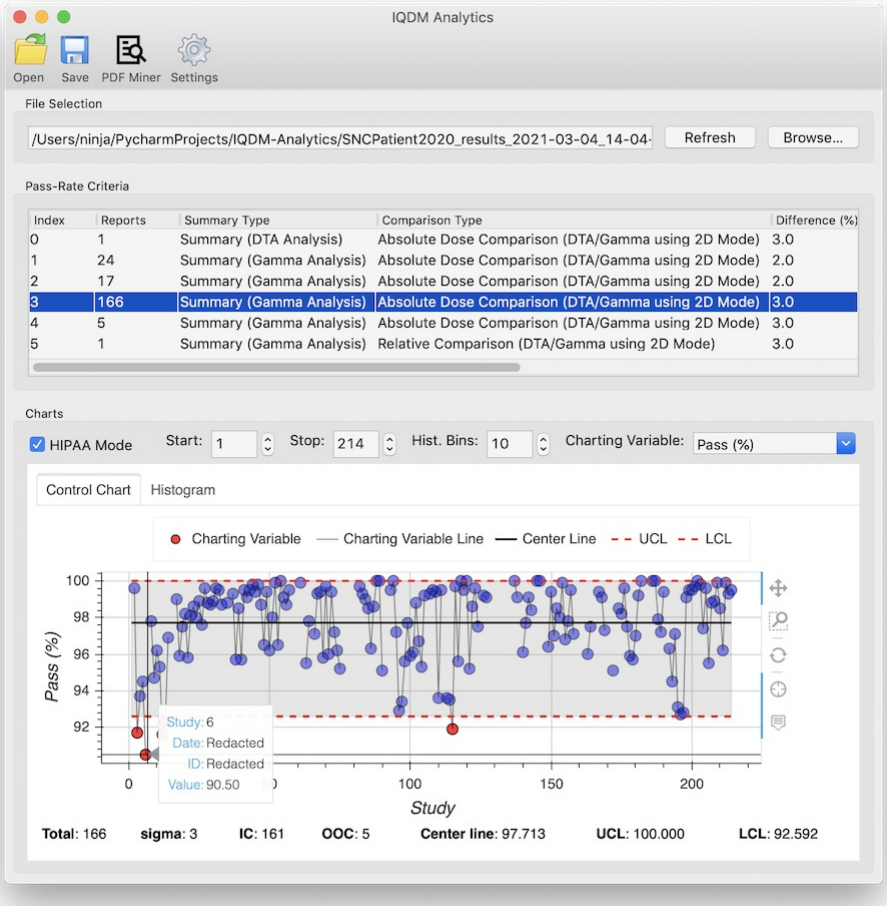
TWO

USER MANUAL

This application is part of the IMRT QA Data Mining (IQDM) project for the AAPM IMRT Working Group (WGIMRT).

2.1 Introduction

IQDM Analytics is a desktop application designed to make IQDM-PDF more user friendly. IQDM-PDF is a python library used to mine data from IMRT QA PDF reports for the purpose of generating control charts, as recommended by AAPM TG-218.



2.2 Usage

The easiest way to use this application is to download an executable from the attachments in the latest release of IQDM Analytics.

Once you've launched the application, click on the PDF Miner icon in the toolbar. From there you can select a directory to scan and another to store a CSV file of mined data. Once this is complete (or if you already have an IQDM-PDF CSV file), click on Open in the toolbar of main window to import the CSV file.

The visuals are created with Bokeh and can be exported to HTML, SVG, or PNG files. Clicking the Save icon in the toolbar will open a window allowing you to apply temporary visual customizations prior to export. Alternatively, you can edit these visuals in Settings to store the changes permanently.

2.3 Supported Vendors

IQDM-PDF currently supports the following IMRT QA vendors / reports:

- Sun Nuclear: SNC Patient
- Scandidos: Delta4
- PTW: Verisoft

2.4 Methods

2.4.1 PDF Mining

Generally speaking, the text from IMRT QA reports is extracted and sorted into boxes with coordinates, using pdfminer.six. Then IQDM-PDF searches for keywords to locate boxes containing data of interest. For more details, see the IQDM-PDF: How It Works page.

Although IQDM-PDF has very thorough testing, it is prudent for users to manually inspect the CSV file generated. If you find an error, please submit an issue with IQDM-PDF. If you provide an anonymized report reproducing the error, it can be included in the automated tests.

2.4.2 Data Parsing

Output from IQDM-PDF will be sorted in the following order:

1. Patient Name & ID (& Plan ID/Name/SOPInstanceUID if available)
2. Analysis parameters (*e.g.*, dose, distance, threshold, etc.)
3. Measurement date & time (or report date)

If multiple reports are found with this sorting, IQDM Analytics can be customized to select either the first or last report (by file creation timestamp), or be set to the min, mean, or max value (calculated per charting variable). See "Duplicate Value Policy" in Settings.

2.4.3 Control Charts

A control chart is simply a plot of chronological data with a center line and control limits (upper and lower). The center line is the mean value of all points. IQDM Analytics calculates a 2-point moving-range,

$$\overline{mR} = \frac{1}{n-1}\sum_{i=2}^{n}|x_i - x_{i-1}|.$$

Control limits (CL) are calculated with

$$CL = \overline{x} \pm 3\frac{\overline{mR}}{1.128},$$

where 3 is the number of standard deviations, which can be customized in Settings. Since the chart is based on a 2-point moving-range, 1.128 is used (*i.e.*, the value of d_2). Note that control limits are bounded if the population it describes also is bounded. For example, the UCL of a gamma pass-rate will not exceed 100%.

The control chart in the main view uses the following acronyms:

- **IC**: In Control

- **OOC**: Out Of Control
- **UCL**: Upper Control Limit
- **LCL**: Lower Control Limit

2.5 CSV Parsing

If you are opening a CSV file generated by IQDM-PDF, its format will be automatically detected and loaded based on instructions from its matching JSON file found in `~/Apps/iqdm_analytics/csv_templates`. If you develop your own data mining script, you can still use IQDM-Analytics if you create a CSV template (JSON formatted). Below is a simple example:

```
{
  "columns": [
    "patient",
    "plan",
    "field id",
    "image type",
    "date",
    "DD(%)",
    "DTA(mm)",
    "Threshold(%)",
    "Gamma Pass Rate(%)"
  ],
  "analysis_columns": {
    "uid": [0, 1, 2],
    "date": 4,
    "criteria": [5, 6, 7],
    "y": [
      {
        "index": 8,
        "ucl_limit": 100,
        "lcl_limit": 0
      }
    ]
  }
}
```

2.5.1 columns

This is a list of columns to be imported, their values must match EXACTLY with the column header in your CSV.

2.5.2 analysis_columns: uid

This is a list of column indices, that when combined, create an ID that is unique to an "observation" or "case". This is used to catch duplicate reports being read. You may specify as column headers instead of indices.

2.5.3 analysis_columns: date

The assigned date for chronological sorting is based on this column index. You may specify as a column header name instead of a column index.

2.5.4 analysis_columns: criteria

These indices are used to "widen" the data (*i.e.*, separate your reports by pass-rate criteria). Generally speaking, this is really a list of independent variables.

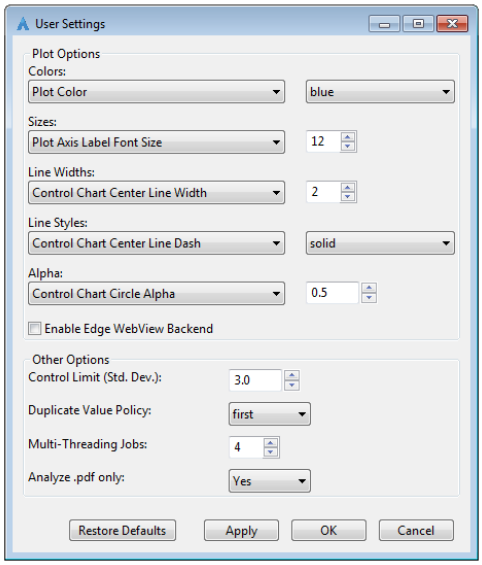
2.5.5 analysis_columns: y

This is a list of dependent variables available for charting. Each item in this list must be a dictionary with the keys `index`, `ucl_limit`, `lcl_limit`. If the charting variable has no bounded control limits, or you do not know them, set the limit values to `null` (*e.g.*, `"ucl_limit": null`). The value for `index` may also be a column header instead of a column index.

2.6 Settings

The Settings window allows you to customize plot visualizations such as colors, widths/sizes, line styles, and transparency (alpha). Additionaly, there are the following options:

- Control Limit standard deviations
  - Set the number of standard deviations for UCL/LCL calculations
- Duplicate Value Policy
  - If multiple reports are found for a given patient/date/ID, use either 'first', 'last', 'min', 'mean', or 'max' value
  - If "Enable Duplicate Detection" is unchecked, all reports will be considered unique observations / cases.
- Multi-Threading Jobs
  - IQDM-PDF supports multi-threading, set the number of jobs used for PDF parsing
- Analyze .pdf only
  - IQDM-PDF looks only at .pdf files by default, allow it to try parsing any file



2.7 Windows Users

The framework used to build this application (wxPython) leverages your operating system's web viewer to render web pages (such as the Bokeh visuals in this application). Unfortunately, Windows still uses Internet Explorer (IE) emulation. This means there is no drag functionality (so no pan or zoom). These features can be recovered if you install Microsoft Edge Beta. If this is installed, you should be able to check "Enable Edge WebView Backend" in Settings. Note that it is much slower to initialize, but you can pan, zoom, and show/hide plot components when clicking on legend items.

Alternatively, you can export your chart as html or navigate to `~/Apps/iqdm_analytics/temp` where the last chart you generated will live as an html file until you render a new one in IQDM Analytics. Then open the file in your browser of choice for full interactive functionality.

2.8 Local File Storage

IQDM Analytics will create the directory `~/Apps/iqdm_analytics`. Your options are stored here as a hidden file *.options*. This directory also contains `csv_templates`, `logs`, and `temp` directories. The `csv_templates` contains instructions for CSV parsing - stored as JSON files. The `logs` contains a `iqdma.log` file if any python errors have been caught. This file will be helpful when reporting any issues. The `temp` directory is currently only used for html file storage on Windows.

2.9 PyInstaller

The executables for IQDM Analytics are generated with PyInstaller, which basically packages a full version of python and necessary libraries. When you run the executable, it unpacks into a temp directory with a location depending on your OS, but starts with `_MEIxxxxxx` where `xxxxxx` is a random number. If the application crashes or you kill the application, just note that this folder will not be automatically purged.

CHAPTER

THREE

IQDM ANALYTICS

3.1 Data Table

A class to sync a data object and list_ctrl

class iqdma.data_table.**DataTable**(*list_ctrl: wx.ListCtrl*, *data: Optional[dict] = None*, *columns: Optional[list] = None*, *widths: Optional[list] = None*, *formats: Optional[list] = None*)

Bases: `object`

Helper class for `wx.ListCtrl`

Init DataTable class

Parameters

- **list_ctrl** (*wx.ListCtrl*) – the list_ctrl in the GUI to be updated with data in this class
- **data** (*dict*) – data should be formatted in a dictionary with keys being the column names and values being lists
- **columns** (*list*) – the keys of the data object to be visible in the list_ctrl
- **widths** (*list*) – optionally specify the widths of the columns
- **formats** (*list*) – optionally specify wx Format values (e.g., wx.LIST_FORMAT_LEFT)

append_row(*row: list*, *layout_only: bool = False*)

Add a row of data

Parameters

- **row** (*list*) – data ordered by self.columns
- **layout_only** (*bool*) – If true, only add row to the GUI

append_row_to_data(*row: list*)

Add a row of data to self.data

Parameters **row** (*list*) – data ordered by self.columns

clear()

Delete all data in self.data and clear the table view

property column_count: int

Number of columns

Returns Length of `columns`

Return type int

property data_for_csv: list
Iterate through data to get a list of csv rows

Returns list of rows. Each row is a list of column data

Return type list of lists

data_to_list_of_rows() → list
Convert data into a list of rows as needed for list_ctrl

Returns data in the format of list of rows

Return type list

delete_all_rows(*layout_only: bool = False*, *force_delete_data: bool = False*)
Clear all data from data and the layout view

Parameters

- **layout_only** (*bool*) – If True, do not remove the row from self.data
- **force_delete_data** (*bool*) – If true, force deletion even if layout is not set

get_csv_rows() → list
Convert data to a list of strings for CSV writing

Returns Each item is a str for a CSV file

Return type list of str

get_data_in_original_order() → dict
Get data in the order it was original set

Returns keys are column names with voalues of row data

Return type dict

get_row(*row_index: int*) → list
Get a row of data from self.data with the given row index

Parameters **row_index** (*int*) – retrieve all values from row with this index

Returns values for the specified row

Return type list

get_value(*row_index: int*, *column_index: int*)
Get a specific table value with a column name and row index

Parameters

- **row_index** (*int*) – retrieve value from row with this index
- **column_index** (*int*) – retrieve value from column with this index

Returns value corresponding to provided indices

Return type any

property has_data: bool
Check if there are any rows of data

Returns True if row_count > 0

Return type bool

increment_index(*evt: Optional[wx.Event] = None*, *increment: Optional[int] = None*)
Increment the ListCtrl selection with an event or fixed increment

Parameters

- **evt** (*wx.Event*) – An event with a GetKeyCode method
- **increment** (*int*) – If no event is passed, use a fixed index increment

property keys: list
Column names

Returns A copy of columns

Return type list

property row_count: int
Number of rows

Returns Length of first column in data

Return type int

property selected_row_data: list
Row data from the current selection in wx.ListCtrl

Returns row data of the currently selected row in the GUI

Return type list

property selected_row_index: list
Get the indices of selected rows in wx.ListCtrl

Returns List of indices

Return type list

set_column_width(*index: int*, *width: int*)
Change the column width in the view

Parameters

- **index** (*int*) – index of column
- **width** (*int*) – the specified width

set_column_widths(*auto: bool = False*)
Set all widths in layout based on widths

Parameters **auto** (*bool*) – Use wx.LIST_AUTOSIZE_USEHEADER rather than widths

set_data(*data: dict*, *columns: list*, *formats: Optional[list] = None*, *ignore_layout: bool = False*)
Set data and update layout

Parameters

- **data** (*dict*) – data should be formatted in a dictionary with keys being the column names and values being lists
- **columns** (*list*) – the keys of the data object to be visible in the list_ctrl
- **formats** (*list*) – optionally specify wx Format values (e.g., wx.LIST_FORMAT_LEFT)
- **ignore_layout** (*bool*) – If true, do not update layout

set_data_in_layout()
Set data in layout from data

set_layout_columns()
Clear layout and re-add columns

sort_table(*evt: wx.EVT_LIST_COL_CLICK*)
Sort the data based on the clicked column header

Parameters **evt** (*wx.EVT_LIST_COL_CLICK*) – Event from a ListCtrl column header click

3.2 Importer

Import output from IQDM-PDF

class iqdma.importer.**CSVParser**(*json_file_path: str*)
Bases: object

Import CSV Template from JSON

Initialization of CSVParser

Parameters **json_file_path** (*str*) – file path to JSON file containing CSV template info

get_index(*value: str*) → int
If value is a string, return its index of columns

Parameters **value** (*str, int*) – any value

Returns If value from analysis_columns is a string, return its index

Return type int

set_values_to_index()
If values are str, set to column index

class iqdma.importer.**ReportImporter**(*report_file_path: str*, *parser: str*, *duplicate_detection: bool*)
Bases: object

Class to import IQDM-PDF CSV output

Initialize ReportImporter

Parameters

- **report_file_path** (*str*) – File path to CSV output from IQDM-PDF
- **parser** (*str*) – The parser used to generate the report. Either 'SNCPatient2020', 'SNCPatientCustom', 'Delta4', 'Verisoft', 'VarianPortalDosimetry'
- **duplicate_detection** (*bool*) – If true, apply a multi_value policy from options

property charting_options: list
Column names of y-axis options

Returns Column names from analysis_columns['y']

Return type list

property criteria_col: list
Column names of analysis criteria options

Returns Column names from analysis_columns['criteria']

Return type list

property lcl
Lower Control Limit minimums

Returns keys are column names, values are minimum LCL values (or None)

Return type dict

remove_non_numeric(*val: str*) → float
Remove all non-numeric characters, convert to float, use to highjack `dtype` in `widen_data`

Parameters **val** (*str*) – Any string

Returns `val` converted into a float

Return type float

property ucl: dict
Upper Control Limit caps

Returns keys are column names, values are maximum UCL values (or None)

Return type dict

property uid_col: list
Column names, when combined create a UID

Returns Column names from `analysis_columns['uid']`

Return type list

iqdma.importer.**copy_default_csv_templates**()
Copy default JSON file form resources/csv_templates

iqdma.importer.**create_csv_template**(*parser: IQDMPDF.parsers.generic.ParserBase*)
Write a CSV_TEMPLATE to JSON

Parameters **parser** (*ParserBase*) – a parser from IQDMPDF

iqdma.importer.**create_default_parsers**()
Generate CSV_TEMPLATE JSON files from IQDMPDF, if it doesn't exist

iqdma.importer.**import_csv_templates**() → dict
Import CSV Templates

Returns keys are parser names and values are `CSVParser` objects. If a default parser is missing from `CSV_TEMPLATES_DIR`, load directly from `IQDMPDF`

Return type dict

3.3 Stats

Modified DVHA-Stats for IQDM-PDF output

class iqdma.stats.**ControlChart**(*y*, *std=3*, *ucl_limit=None*, *lcl_limit=None*, *x=None*, *range=None*)
Bases: `object`

Calculate control limits for a standard univariate Control Chart"

Parameters

- **y** (*list, np.ndarray*) – Input data (1-D)
- **std** (*int, float, optional*) – Number of standard deviations used to calculate if a y-value is out-of-control.
- **ucl_limit** (*float, optional*) – Limit the upper control limit to this value
- **lcl_limit** (*float, optional*) – Limit the lower control limit to this value
- **range** (*tuple, list, ndarray*) – 2-item object containing start and end index of `y`

Initialization of a ControlChart

property avg_moving_range
Avg moving range based on 2 consecutive points

Returns Average moving range. Returns NaN if arr is empty.

Return type np.ndarray, np.nan

property center_line
Center line of charting data (i.e., mean value)

Returns Mean value of y with np.mean() or np.nan if y is empty

Return type np.ndarray, np.nan

property chart_data
JSON compatible dict for chart generation

Returns Data used for Histogram visuals. Keys include 'x', 'y', 'out_of_control', 'center_line', 'lcl', 'ucl'

Return type dict

property control_limits
Calculate the lower and upper control limits

Returns

- **lcl** (*float*) – Lower Control Limit (LCL)
- **ucl** (*float*) – Upper Control Limit (UCL)

property out_of_control
Get the indices of out-of-control observations

Returns An array of indices that are not between the lower and upper control limits

Return type np.ndarray

property out_of_control_high
Get the indices of observations > ucl

Returns An array of indices that are greater than the upper control limit

Return type np.ndarray

property out_of_control_low
Get the indices of observations < lcl

Returns An array of indices that are less than the lower control limit

Return type np.ndarray

property sigma
UCL/LCL = center_line +/- sigma * std

Returns sigma or np.nan if arr is empty

Return type np.ndarray, np.nan

property x_ranged: list
Return `x` within `range`

Returns `x` data from `range[0]` to `range[1]`

Return type list

property y_ranged
Return y within range

Returns y data from range[0] to range[1]

Return type list

class iqdma.stats.**IQDMStats**(*report_file_path: str*, *charting_column: str*, *multi_val_policy: str*, *duplicate_detection: bool*, *parser: str*)

Bases: object

Modified DVHAStats class for IQDM-PDF output

Initialize IQDMStats

Parameters

- **report_file_path** (*str*) – File path to CSV output from IQDM-PDF
- **charting_column** (*str*) – Column of y-axis data
- **multi_val_policy** (*str*) – Duplicate value policy from options
- **duplicate_detection** (*bool*) – If true, apply a multi_value policy from options
- **parser** (*str*) – CSV format

get_index_by_var_name(*var_name: str*)
Get the variable index by var_name

Parameters **var_name** (*int, str*) – The name (str) or index (int) of the variable of interest

Returns The column index for the given var_name

Return type int

get_index_description() → tuple
Get a dict of data and columns for *DataTable*

Returns

- *dict* – Keys are column names with values being a list of values
- *list* – Column names in order to be displayed

univariate_control_chart(*var_name: str*, *std: float = 3*, *ucl_limit: Optional[float] = None*, *lcl_limit: Optional[float] = None*, *range: Optional[tuple] = None*)
Calculate control limits for a standard univariate Control Chart

Parameters

- **var_name** (*str, int*) – The name (str) or index (int) of teh variable to plot
- **std** (*int, float, optional*) – Number of standard deviations used to calculate if a y-value is out-of-control
- **ucl_limit** (*float, optional*) – Limit the upper control limit to this value
- **lcl_limit** (*float, optional*) – Limit the lower control limit to this value
- **range** (*tuple, list, ndarray*) – 2-item object containing start and end index of data

Returns stats.ControlChart class object

Return type *stats.ControlChart*

univariate_control_charts(***kwargs*)
Calculate Control charts for all variables

Parameters **kwargs** (*any*) – See univariate_control_chart for keyword parameters

Returns ControlChart class objects stored in a dictionary with var_names and indices as keys (can use var_name or index)

Return type dict

property variable_count
Number of variables in data

Returns Number of columns in data

Return type int

iqdma.stats.**avg_moving_range**(*arr*, *nan_policy='omit'*)
Calculate the average moving range (over 2-consecutive point1)

Parameters

- **arr** (*array-like (1-D)*) – Input array. Must be positive 1-dimensional.
- **nan_policy** (*str, optional*) – Value must be one of the following: {'propagate', 'raise', 'omit'} Defines how to handle when input contains nan. The following options are available (default is 'omit'): 'propagate': returns nan 'raise': throws an error 'omit': performs the calculations ignoring nan values

Returns Average moving range. Returns NaN if arr is empty

Return type np.ndarray, np.nan

iqdma.stats.**process_nan_policy**(*arr*, *nan_policy*)
Calculate the average moving range (over 2-consecutive point1)

Parameters

- **arr** (*array-like (1-D)*) – Input array. Must be positive 1-dimensional.
- **nan_policy** (*str*) – Value must be one of the following: {'propagate', 'raise', 'omit'} Defines how to handle when input contains nan. The following options are available (default is 'omit'): 'propagate': returns nan 'raise': throws an error 'omit': performs the calculations ignoring nan values

Returns Input array evaluated per nan_policy

Return type np.ndarray, np.nan

iqdma.stats.**remove_nan**(*arr*)
Remove indices from 1-D array with values of np.nan

Parameters **arr** (*np.ndarray (1-D)*) – Input array. Must be positive 1-dimensional.

Returns arr with NaN values deleted

Return type np.ndarray

3.4 Utilities ported from DVHA-Stats

Common functions for DVHA-Stats. Copied to limit required libraries.

iqdma.utilities_dvha_stats.**apply_dtype**(*value*, *dtype*)
Convert value with the provided data type

Parameters

- **value** (*any*) – Value to be converted
- **dtype** (*function, None*) – python reserved types, e.g., int, float, str, etc. However, dtype could be any callable that raises a ValueError on failure.

Returns The return of dtype(value) or numpy.nan on ValueError

Return type any

iqdma.utilities_dvha_stats.**csv_to_dict**(*csv_file_path*, *delimiter=','*, *dtype=None*, *header_row=True*)
Read in a csv file, return data as a dictionary

Parameters

- **csv_file_path** (*str*) – File path to the CSV file to be processed.
- **delimiter** (*str*) – Specify the delimiter used in the csv file (default = ',')
- **dtype** (*callable, type, optional*) – Optionally force values to a type (e.g., float, int, str, etc.).
- **header_row** (*bool, optional*) – If True, the first row is interpreted as column keys, otherwise row indices will be used

Returns CSV data as a dict, using the first row values as keys

Return type dict

iqdma.utilities_dvha_stats.**dict_to_array**(*data*, *key_order=None*)
Convert a dict of data to a numpy array

Parameters

- **data** (*dict*) – Dictionary of data to be converted to np.array.
- **key_order** (*None, list of str*) – Optionally the order of columns

Returns A dictionary with keys of 'data' and 'columns', pointing to a numpy array and list of str, respectively

Return type dict

iqdma.utilities_dvha_stats.**get_sorted_indices**(*list_data*)
Get original indices of a list after sorting

Parameters **list_data** (*list*) – Any python sortable list

Returns list_data indices of sorted(list_data)

Return type list

iqdma.utilities_dvha_stats.**import_data**(*data*, *var_names=None*)
Generalized data importer for np.ndarray, dict, and csv file

Parameters

- **data** (*numpy.array, dict, str*) – Input data (2-D) with N rows of observations and p columns of variables. The CSV file must have a header row for column names.

- **var_names** (*list of str, optional*) – If data is a numpy array, optionally provide the column names.

Returns A tuple: data as an array and variable names as a list

Return type np.ndarray, list

iqdma.utilities_dvha_stats.**is_numeric**(*val*)
Check if value is numeric (float or int)

Parameters **val** (*any*) – Any value

Returns Returns true if float(val) doesn't raise a ValueError

Return type bool

iqdma.utilities_dvha_stats.**sort_2d_array**(*arr*, *index*, *mode='col'*)
Sort a 2-D numpy array

Parameters

- **arr** (*np.ndarray*) – Input 2-D array to be sorted
- **index** (*int, list*) – Index of column or row to sort arr. If list, will sort by each index in the order provided.
- **mode** (*str*) – Either 'col' or 'row'

CHAPTER

FOUR

CREDITS

4.1 Development Lead

- Dan Cutright, University of Chicago Medicine

4.2 Contributors

- Marc Chamberland, University of Vermont Health Network
- Serpil Kucuker Dogan, Northwestern Medicine
- Mahesh Gopalakrishnan, Northwestern Medicine
- Aditya Panchal, AMITA Health
- Michael Snyder, Beaumont Health

CHAPTER

FIVE

CHANGE LOG FOR IQDM-ANALYTICS

5.1 v0.1.9 (2021.05.17)

- Clean charting values of non-numerical characters on import
- Prevent crash if only one UID columns is defined in CSV Template
- Added CSV template for IBA MyQA

5.2 v0.1.8 (2021.03.28)

- Implement CSV parsing with JSON templates, allowing for customization
- Option to disable duplicate report detection

5.3 v0.1.7 (2021.03.14)

- Added Control Chart using all data
- MS Edge Support for Windows
- IQDM-PDF bump to v0.3.0

5.4 v0.1.6 (2021.03.13)

- Improved date parsing (IQDM-PDF bump to v0.2.9)

5.5 v0.1.5 (2021.03.10)

- Last release before Change Log implemented

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

INDEX

L

M

O

P

R

S

U

V

X

Y