DQuality Documentation

Release 0.0.2

Argonne National Laboratory

Jun 21, 2017

Contents

DQuality is an open-sourced Python package to perform data integrity and quality assessment at the Advanced Photon Source.

This guide is maintained on GitHub.

Data Quality assurance is an essential task to provide consistency, accuracy, reliability and reproducibility of scientific results *[C1]*, *[C2]*, *[C3]*, *[C4]*.

DQuality is a python toolbox to asses the quality of raw and processed data collected at the Advanced Photon Source (APS).

DQuality verifies that all experiment components EPICS Process Variables (PVs) like motor positions, storage ring parameters etc. are valid and their values are within a predefined range.

DQuality verifies that the raw data files saved in the Data Exchange hdf5 file format contain a valid data and meta-data structure and that existing dependencies between the data and meta-data structure are correct.

DQuality provides live monitoring of the above functionalities for all raw data created in the active directory.

CHAPTER 1

Install

This section covers the basics of how to download and install DQuality.

Contents:

Pre-requisites

Each of the verifier requires a parameter configuration file. The configuration files must define schemas and verifier's properties and be located in the user account home directory under a .dquality/default folder.

- `~/.dquality/default/dqconfig.ini`

The schemas should be saved under "*default/schemas*" folder and contain the following files:

- `~/.dquality/default/schemas/pvs.json` containing the list of Process variable (PV) of your beamline PVs with their acceptable value range.
- `~/.dquality/default/schemas/tags.json` containing the list valid HDF file tags, attributes and array dimentions.
- `~/.dquality/default/schemas/dependencies.json` containing the list of valid relation among data sets in the same HDF file.
- `~/.dquality/default/schemas/limits.json` containing the threshold values for the quality check calculations.

- `~/.dquality/default/schemas/data_tags.json` containing the dictionary of hdf tags of data sets by data type.
- `~/.dquality/default/schemas/quality_checks.json` containing the dictionary of quality check calculations. The keys are the slice verification methods, values are lists of stat methods.

Different instruments generating data that require different sets of configuration files can be configured as ~/.dquality/instrument1, ~/.dquality/instrument2 etc. with the same schemas subfolder structure.

Installing from source

Clone the DQuality from GitHub repository:

```
git clone https://github.com/bfrosik/data-quality.git DQuality
```

then:

```
cd DQuality
python setup.py install
```

Installing from Conda/Binstar (coming soon)

First you must have Conda installed, then open a terminal or a command prompt window and run:

```
conda install -c ....
```

Updating the installation (coming soon)

Data Management is an active project, so we suggest you update your installation frequently. To update the installation run in your terminal:

```
conda update -c ....
```

For some more information about using Conda, please refer to the docs.

CHAPTER 2

Cofiguration

This describes configuration of mandatory and optional parameters. The common section depicts parameters used by all verifiers. A list of additional parameters specific for each verifier follows.

common configuration

- 'log_file':

optional, a log file name including path. If not specified, the log file default.log will be created in the running directory

- 'time_zone':

optional, time zone that will be displayed as part of timestamp in log file. If not specified, it defaults to 'America/Chicago'

pv verifier

- 'pv_file':

mandatory, json file name including path that specifies pv requirements

hdf verifier

- 'schema':

mandatory, json file name including path that specifies hdf tags requirements

- 'verification_type':

optional. Currently the software supports 'hdf_structure' and 'hdf_tags' types. If not specified it defaults to 'hdf_structure' type. When configured to 'hdf_structure' the tags and attributes specified in 'schema' file will be evaluated. If configured to 'hdf_tags', only presence of the tags specified in 'schema' file will be evaluated.

dependency verifier

- 'dependencies':

mandatory, json file name including path that specifies hdf tags dependency requirements

data verifier

- 'limits':

mandatory, json file name including path that specifies limits used in quality checks 'quality_checks': mandatory, json file name including path that lists all quality methods that will be used to validate the data.

- 'file_type':

optional, data file type. Currently the software supports FILE_TYPE_GE and FILE_TYPE_HDF formats. If not specified it defaults to FILE_TYPE_HDF format.

- 'data_tags':

required for hdf type file. json file name including path that maps hdf tags to data types ('data', 'data_dark','data_white') that will be verified.

- 'report_type':

optional, defines report specifics. Currently the software supports 'none', 'full', and 'errors' types. If not specified it defaults to 'full' type. If the type is 'none', no report file will be created; if the type is 'errors', only the bad frames will be reported; and for the 'full' report type all frames and check results are reported.

- 'report_dir':

optional, a directory where report files will be located. If not configured, the report files are created along the data files.

- 'feedback_type':

optional, defines a real time feedback when validating data. For data verifier it should not be set, or set to "none'

monitor

- 'limits':

mandatory, json file name including path that specifies limits used in quality checks 'quality_checks': mandatory, json file name including path that lists all quality methods that will be used to validate the data.

- 'file_type':

optional, data file type. Currently the software supports FILE_TYPE_GE and FILE_TYPE_HDF formats. If not specified it defaults to FILE_TYPE_HDF format.

- 'data_tags':

required for hdf type file. json file name including path that maps hdf tags to data types ('data', 'data_dark', 'data_white') that will be verified.

- 'report_type':

optional, defines report specifics. Currently the software supports 'none', 'full', and 'errors' types. If not specified it defaults to 'full' type. If the type is 'none', no report file will be created; if the type is 'errors', only the bad frames will be reported; and for the 'full' report type all frames and check results are reported.

- 'report_dir':

optional, a directory where report files will be located. If not configured, the report files are created along the data files.

- 'feedback_type':

optional, defines a real time feedback when validating data. For data verifier it should not be set, or set to "none'

accumulator

- 'limits':

mandatory, json file name including path that specifies limits used in quality checks 'quality_checks': mandatory, json file name including path that lists all quality methods that will be used to validate the data.

- 'quality_checks':

mandatory, json file name including path that lists all quality methods that will be used to validate the data.

- 'report_type':

optional, defines report specifics. Currently the software supports 'none', 'full', and 'errors' types. If not specified it defaults to 'full' type. If the type is 'none', no report file will be created; if the type is 'errors', only the bad frames will be reported; and for the 'full' report type all frames and check results are reported.

- 'feedback_type':

optional, defines a real time feedback when validating data. For data verifier it should not be set, or set to "none'

real_time verifier

- 'limits':

mandatory, json file name including path that specifies limits used in quality checks 'quality_checks': mandatory, json file name including path that lists all quality methods that will be used to validate the data.

- 'quality_checks':

mandatory, json file name including path that lists all quality methods that will be used to validate the data.

- 'report_type':

optional, defines report specifics. Currently the software supports 'none', 'full', and 'errors' types. If not specified it defaults to 'full' type. If the type is 'none', no report file will be created; if the type is 'errors', only the bad frames will be reported; and for the 'full' report type all frames and check results are reported.

- 'feedback_type':

optional, a list that defines a real time feedback when validating data. Currently the software supports 'log', 'console', and 'pv'. If the list contains 'console', the software will print the failed verification results in the real time; if the list contain 'log', the failed results will be logged.

- 'detector':

mandatory, specifies EPICS Area Detector prefix, as defined in the area detector configuration

- ‘detector_basic’:

mandatory, specifies EPICS Area Detector second prefix that is used for the basic PVs, as defined in the area detector configuration

- ‘detector_image’:

mandatory, specifies EPICS Area Detector second prefix that is used for the image PVs, as defined in the area detector configuration

- ‘no_frames’:

mandatory, number of frames that the real time verifier will evaluate. It will run undefinately when set to -1.

CHAPTER 3

API reference

DQuality provides the following funtionalities:

- "*PV*": Before data collection start, verify that the experiment setup PVs, i.e. all required setup data, are valid and their values are within a predefined range.
- "*Hdf*": verify the correctness of the data and meta-data structure in an hdf5 file.
- "*Hdf Dependencies*": verify dependencies between the data and meta-data structure in an hdf5 file.
- "*Data*": verify the quality of the data after data is collected in a file. A set of QC functions is provided to assess the image quality against different criteria (mean, dynamic range, structural similarity, multi-scale structural similarity, visual information fidelity, most apparent distortion, etc.) *[C4]*. The resulting "*limit*", related to the quantitive QC functions, defines whether the data is of good or poor quality. The limit values, at first, are set by the research/tests with trial data sets. The QC function "*limit*" range will eventually be learned by implementing a learning mechanism. Any calculated "*result*" quality parameter is stored, in the case of hdf format, in the file itself with a corresponding tag. If the data file format supports only raw data (no meta-data), the quality parameter results are stored in a separate file with a name corresponding to the data file.
- "*Monitor*", "*Monitor_polling*": monitor the active data collection directory and run "*Data*" on each new file.
- "*Accumulator*": monitor the active data collection directory where each new file is part of the same data set.
- "*Realtime*": verifies the quality of the active EPICS Channel Access data in a real time.
- "*Check*": provides a wrapper to "*PV*", "*Hdf*", "*Hdf Dependencies*", "*Data*", "*Monitor*", and "*Accumulator*".
- "*realtime.Check*": provides a wrapper to "*Realtime*".

DQuality Modules:

dquality.realtime.real_time

dquality.realtime.check

dquality.accumulator

dquality.check

dquality.data

dquality.hdf

Please make sure the installation *Pre-requisites* are met.

This file contains verification functions related to the file structure. It reads configuration parameters "*schema_type*" and "*schema*" to determine first which kind of file verification is requested, and a schema that defines mandatory parameters. If any of the parameters is not configured, it is assumed no file structure verification is requested.

Functions:

init(config)	This function initializes global variables.
report_items(list, text1, text2, logger)	This function takes a list and strings.
verify(conf, file)	This is the main function called when the structureverifier application starts.
structure(file, required_tags, logger)	This method is used when a file of hdf type is given.
tags(file, required_tags, logger)	This method is used when a file of hdf type is given.

dquality.hdf.**init**(*config*)

This function initializes global variables. It gets values from the configuration file, evaluates and processes the values. If mandatory file or directory is missing, the script logs an error and exits.

Parameters **config** (*str*) – configuration file name, including path

Returns

- **logger** (*Logger*) – logger instance
- **tags** (*dictionary*) – a dictionary containing tag and attributes values read from the configured 'schema' file

dquality.hdf.**report_items**(*list*, *text1*, *text2*, *logger*)

This function takes a list and strings. If the list is not empty it prints the two string parameters as a title, and prints formatted output for each item in a list.

Parameters

- **list** (*list*) – A list of items
- **text1** (*str*) – A title that will be printed if the list is not empty

- **text2** (*str*) – An optional part of title that will be printed if the list is not empty
- **logger** (*Logger*) – a Logger instance

Returns *None*

`dquality.hdf.`**`verify`**(*conf*, *file*)

This is the main function called when the structureverifier application starts. It reads the configuration file for "*verification_type*" to verify "*hdf_structure*" or "*hdf_tags*".

Parameters

- **conf** (*str*) – configuration file name, including path
- **file** (*str*) – File Name to verify including path

Returns *boolean*

`dquality.hdf.`**`tags`**(*file*, *required_tags*, *logger*)

This method is used when a file of hdf type is given. All tags from the hdf file are added in the filetags list. Then the schema is evaluated for tags. With each tag discovered it checks whether there is matching tag in the filetags list. If a tag is missing, the function exits with False. Otherwise, it will return True.

Parameters

- **file** (*str*) – File Name including path
- **schema** (*str*) – Schema file name
- **logger** (*Logger*) – a Logger instance

Returns

- *True if verified*
- *False if not verified*

`dquality.hdf.`**`structure`**(*file*, *required_tags*, *logger*)

This method is used when a file of hdf type is given. All tags and array dimensions are verified against a schema. (see `schemas/tags.json` example file).

Parameters

- **file** (*str*) – File Name including path
- **schema** (*str*) – Schema file name
- **logger** (*Logger*) – a Logger instance

Returns *None*

dquality.hdf_dependency

Please make sure the installation *Pre-requisites* are met.

This module verifies that each of the PVs listed in the configuration file exist and their values are set within the predefined range.

The results will be reported in a file (printed on screen for now). An error will be reported back to UI via PV.

Functions:

init(config)	This function initializes global variables.
verify(conf, file)	This function reads the json "*dependencies*" file from the `dqconfig.ini` file.
verify_list(file, list, relation, logger)	This function takes an hd5 file, a list of tags (can be extended) and a relation between the list members.
find_value(tag, dset)	This function takes tag parameter and a corresponding dataset from the hd5 file.

dquality.hdf_dependency.**init**(*config*)

This function initializes global variables. It gets values from the configuration file, evaluates and processes the values. If mandatory file or directory is missing, the script logs an error and exits.

Parameters **config** (*str*) – configuration file name, including path

Returns

- **logger** (*Logger*) – logger instance
- **dep** (*dictionary*) – a dictionary containing dependency values read from the configured 'dependencies' file

dquality.hdf_dependency.**verify**(*conf*, *file*)

This function reads the json "*dependencies*" file from the `dqconfig.ini` file. This file contains dictionary with keys of relations between tags. The value is a list of lists. The relation applies to the tags in inner list respectively. For example if the relation is "*equal*", all tags in each inner list must be equal,The outer list hold the lists that apply the relation. A first element in a inner list is an "*anchor*" element, so all elements are compared to it. This is important for the "*less_than*" type of relation, when the order of parameters is important.

The allowed keys are:

- "*less_than*" - the PV value must be less than attribute value
- "*less_or_equal*" - the PV value must be less than or equal attribute value
- "*equal*" - the PV value must be equal to attribute value
- "*greater_or_equal*" - the PV value must be greater than or equal attribute value
- "*greater_than*" - the PV value must be greater than attribute value

The tag in a "*dependencies*" file can be an hd5 tag, or can have appended keyword "*dim*" and an numeric value indicating axis. If the tag is simple hd5 tag, the verifier compares the value of this tag. If it has the "*dim*" keyword appended, the verifier compares a specified dimension.

Any vakue that does not agree with the configured relation is reported (printed for now). The function returns True if no error was found and False otherwise.

Parameters

- **conf** (*str*) – configuration file name, including path
- **file** (*str*) – File Name to verify including path

Returns *boolean*

dquality.hdf_dependency.**verify_list**(*file*, *list*, *relation*, *logger*)

This function takes an hd5 file, a list of tags (can be extended) and a relation between the list members. First the method creates a tags dictionary from the list of tags. The key is a simple hd5 tag, and the value is a newly created TagValue instance that contains extended tag (or simple tag if simple tag was in the list), and a placeholder for the value. The first tag from the list is retained as an anchor.

The function travers through all tags in the given file. If the tag name is found in the tags dictionary, the `find_value` method is called to retrieve a defined valueu referenced by the tag. The value is then added to the TagValue instance for this tag.

When all tags are processed the function iterates over the tags dictionary to find if the relation between anchor value and other values can be verified. If any relation is not true, a report is printed for this tag, and the function will return `False`. Otherwise the function returns `True`.

Parameters

- **file** (*file*) – an hd5 file to be verified
- **list** (*list*) – list of extended or simple hd5 tags
- **relation** (*str*) – a string specifying the relation between tags in the list
- **logger** (*Logger*) – a Logger instance

Returns *boolean*

dquality.hdf_dependency.**find_value**(*tag*, *dset*)

This function takes tag parameter and a corresponding dataset from the hd5 file. The tag can be a simple hd5 member tag or extended tag. This function assumes that the extended tag is a string containing hd5 tag, a word indicating the tag category (i.e. "*dim*" meaning dimension), and a parameter (i.e. numeric index). In the case of a simple hd5 tag the function returns a value of this member. In the second case the function returns decoded value, in the case of "dim" category, it returns the indexed dimension.

Parameters

- **tag** (*str*) – a simple hd5 tag or extended
- **dset** (*dataset*) – hd5 dataset corresonding to the tag parameter

Returns *value of decoded tag*

dquality.monitor

dquality.pv

Please make sure the installation *Pre-requisites* are met.

This module verifies that each of the PVs listed in the configuration file exist and their values are set within the predefined range.

The results will be reported in a file (printed on screen for now). An error will be reported back to UI via PV.

Functions:

init(config)	This function initializes global variables.
verify(conf)	This function reads the `schemas/pvs.json` as set in the `dqconfig.ini` file.
read(pv_str)	This function returns a Process Variable (PV) value or None if the PV does not exist.
state(value, limit)	This function takes boolean "*value*" parameter and string "limit" parameter that can be either "*True*" or "*False*".

dquality.pv.**init**(*config*)

This function initializes global variables. It gets values from the configuration file, evaluates and processes the values. If mandatory file or directory is missing, the script logs an error and exits.

Parameters **config** (*str*) – configuration file name, including path

Returns

- **logger** (*Logger*) – logger instance
- **pvs** (*dictionary*) – a dictionary containing pvs values and attributes read from the configured 'pv_file' file

dquality.pv.**verify**(*conf*)

This function reads the `schemas/pvs.json` as set in the `dqconfig.ini` file. This file contains dictionary with keys of mandatory process variables. The values is a dictionary of attributes, each attribute being either description, or a verification operation. The verification operation attribute has an operation as a key, and the value is the limit of the PV. The allowed keys are:

•"*less_than*" - the PV value must be less than attribute value

•"*less_or_equal*" - the PV value must be less than or equal attribute value

•"*equal*" - the PV value must be equal to attribute value

•"*greater_or_equal*" - the PV value must be greater than or equal attribute value

•"*greater_than*" - the PV value must be greater than attribute value

•"*state*" - to support boolean PVs. The defined value must be "True" or "False".

Any missing PV (i.e. it can't be read) is an error that is reported (printed for now). Any PV value that is out of limit is an error that is reported (printed for now). The function returns True if no error was found and False otherwise.

Parameters **conf** (*str*) – configuration file name, including path

Returns *boolean*

dquality.pv.**read**(*pv_str*)

This function returns a Process Variable (PV) value or None if the PV does not exist.

Parameters **value** (*str*) – name of the PV

Returns *PV value*

dquality.pv.**state**(*value*, *limit*)

This function takes boolean "*value*" parameter and string "limit" parameter that can be either "*True*" or "*False*". The limit is converted to string and compared with the value. The function returns True if the boolean values are equal, False otherwise.

Parameters

- **value** (*numeric*) – value
- **limit** (*str*) – limit value

Returns *boolean*

CHAPTER 4

Examples

This section contains various DQuality examples. Please make sure the installation *Pre-requisites* are met.

File Accumulator

DQuality Folder Accumulator example. (Download file: `accumulator_check.py`)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ##############################################################################
# Copyright (c) 2015, UChicago Argonne, LLC. All rights reserved.              #
#                                                                              #
# Copyright 2015. UChicago Argonne, LLC. This software was produced            #
# under U.S. Government contract DE-AC02-06CH11357 for Argonne National        #
# Laboratory (ANL), which is operated by UChicago Argonne, LLC for the         #
# U.S. Department of Energy. The U.S. Government has rights to use,            #
# reproduce, and distribute this software.  NEITHER THE GOVERNMENT NOR         #
# UChicago Argonne, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR             #
# ASSUMES ANY LIABILITY FOR THE USE OF THIS SOFTWARE.  If software is          #
# modified to produce derivative works, such modified software should          #
# be clearly marked, so as not to confuse it with the version available       #
# from ANL.                                                                    #
#                                                                              #
# Additionally, redistribution and use in source and binary forms, with        #
# or without modification, are permitted provided that the following           #
# conditions are met:                                                          #
#                                                                              #
#     * Redistributions of source code must retain the above copyright         #
#       notice, this list of conditions and the following disclaimer.          #
#                                                                              #
#     * Redistributions in binary form must reproduce the above copyright      #
#       notice, this list of conditions and the following disclaimer in        #
#       the documentation and/or other materials provided with the             #
```

```
#       distribution.                                                        #
#                                                                            #
#     * Neither the name of UChicago Argonne, LLC, Argonne National          #
#       Laboratory, ANL, the U.S. Government, nor the names of its           #
#       contributors may be used to endorse or promote products derived      #
#       from this software without specific prior written permission.        #
#                                                                            #
# THIS SOFTWARE IS PROVIDED BY UChicago Argonne, LLC AND CONTRIBUTORS        #
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT          #
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS          #
# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UChicago        #
# Argonne, LLC OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,           #
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,       #
# BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;           #
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER           #
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT         #
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN          #
# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE            #
# POSSIBILITY OF SUCH DAMAGE.                                                #
# ############################################################################
import sys
import os
import argparse
from dquality.check import accumulator as acc
from os.path import expanduser

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("instrument", help="instrument name, name should have a␣
↪matching directory in the .dquality folder")
    parser.add_argument("fname", help="folder name to monitor for files")
    parser.add_argument("type", help="data type to be verified (i.e. data_dark, data_
↪white, or data)")
    parser.add_argument("numfiles", help="number of files to monitor for")
    parser.add_argument("repbyfile", help="boolean value defining how the bad indexes␣
↪should be reported.")

    args = parser.parse_args()
    instrument = args.instrument
    fname = args.fname
    dtype = args.type
    num_files = args.numfiles
    report_by_file = args.repbyfile

    home = expanduser("~")
    conf = os.path.join(home, ".dquality", instrument)

    bad_indexes = acc(conf, fname, dtype, num_files, report_by_file)
    return bad_indexes


if __name__ == "__main__":
    main(sys.argv[1:])
```

Data Quality

DQuality Data Quality example. (Download file: `data_check.py`)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ######################################################################
# Copyright (c) 2015, UChicago Argonne, LLC. All rights reserved.      #
#                                                                      #
# Copyright 2015. UChicago Argonne, LLC. This software was produced    #
# under U.S. Government contract DE-AC02-06CH11357 for Argonne National   #
# Laboratory (ANL), which is operated by UChicago Argonne, LLC for the #
# U.S. Department of Energy. The U.S. Government has rights to use,    #
# reproduce, and distribute this software.  NEITHER THE GOVERNMENT NOR #
# UChicago Argonne, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR     #
# ASSUMES ANY LIABILITY FOR THE USE OF THIS SOFTWARE.  If software is  #
# modified to produce derivative works, such modified software should  #
# be clearly marked, so as not to confuse it with the version available   #
# from ANL.                                                            #
#                                                                      #
# Additionally, redistribution and use in source and binary forms, with   #
# or without modification, are permitted provided that the following   #
# conditions are met:                                                  #
#                                                                      #
#     * Redistributions of source code must retain the above copyright    #
#       notice, this list of conditions and the following disclaimer.     #
#                                                                      #
#     * Redistributions in binary form must reproduce the above copyright #
#       notice, this list of conditions and the following disclaimer in   #
#       the documentation and/or other materials provided with the        #
#       distribution.                                                  #
#                                                                      #
#     * Neither the name of UChicago Argonne, LLC, Argonne National       #
#       Laboratory, ANL, the U.S. Government, nor the names of its        #
#       contributors may be used to endorse or promote products derived   #
#       from this software without specific prior written permission.     #
#                                                                      #
# THIS SOFTWARE IS PROVIDED BY UChicago Argonne, LLC AND CONTRIBUTORS     #
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT       #
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS       #
# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UChicago     #
# Argonne, LLC OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,        #
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,    #
# BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;        #
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER        #
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT      #
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN       #
# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE         #
# POSSIBILITY OF SUCH DAMAGE.                                          #
# ######################################################################
"""
Please make sure the installation :ref:`pre-requisite-reference-label` are met.

This script is specific for beamline 32id.

This example takes two mandatory parameters:
instrument: a string defining the detector that will be used. User can enter one of␣
↪these choices:
```

```
'32id_nano', '32id_micro'. The instrument determines the directory to look for a configuration file that will be used.
file: a file to be verified for dependencies according to schema.

This script calls quality_check verifier.

"""
import sys
import os
import argparse
from dquality.check import data as dquality_check
from os.path import expanduser

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("instrument", help="instrument name, name should have a matching directory in the .dquality folder")
    parser.add_argument("fname", help="file name to do the tag dependencies checks on")

    args = parser.parse_args()
    instrument = args.instrument
    fname = args.fname

    home = expanduser("~")
    conf = os.path.join(home, ".dquality", instrument)

    bad_indexes = dquality_check(conf, fname)
    return bad_indexes


if __name__ == "__main__":
    main(sys.argv[1:])
```

HDF File

This module requires the "*schema*" section of the `dqconfig.ini` file to be set. If "*schema*" is not configured, the function returns True, as no verification is needed.

In case the "*verification_type*" of the `dqconfig.ini` file is set, the follow up logic determines, what type of verification should be applied.

Currently, for the HDF schema, both "*hdf_structure*" and "*hdf_tags*" verification types are supported.

DQuality HDF file check example. (Download file: `hdf_check.py`)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ##########################################################################
# Copyright (c) 2015, UChicago Argonne, LLC. All rights reserved.          #
#                                                                          #
# Copyright 2015. UChicago Argonne, LLC. This software was produced        #
# under U.S. Government contract DE-AC02-06CH11357 for Argonne National    #
# Laboratory (ANL), which is operated by UChicago Argonne, LLC for the     #
```

```
# U.S. Department of Energy. The U.S. Government has rights to use,        #
# reproduce, and distribute this software.  NEITHER THE GOVERNMENT NOR     #
# UChicago Argonne, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR         #
# ASSUMES ANY LIABILITY FOR THE USE OF THIS SOFTWARE.  If software is      #
# modified to produce derivative works, such modified software should      #
# be clearly marked, so as not to confuse it with the version available    #
# from ANL.                                                                #
#                                                                          #
# Additionally, redistribution and use in source and binary forms, with    #
# or without modification, are permitted provided that the following       #
# conditions are met:                                                      #
#                                                                          #
#     * Redistributions of source code must retain the above copyright     #
#       notice, this list of conditions and the following disclaimer.      #
#                                                                          #
#     * Redistributions in binary form must reproduce the above copyright  #
#       notice, this list of conditions and the following disclaimer in    #
#       the documentation and/or other materials provided with the         #
#       distribution.                                                      #
#                                                                          #
#     * Neither the name of UChicago Argonne, LLC, Argonne National        #
#       Laboratory, ANL, the U.S. Government, nor the names of its         #
#       contributors may be used to endorse or promote products derived    #
#       from this software without specific prior written permission.      #
#                                                                          #
# THIS SOFTWARE IS PROVIDED BY UChicago Argonne, LLC AND CONTRIBUTORS      #
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT        #
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS        #
# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UChicago      #
# Argonne, LLC OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,         #
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,     #
# BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;         #
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER         #
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT       #
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN        #
# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE          #
# POSSIBILITY OF SUCH DAMAGE.                                              #
# ##########################################################################
import sys
import os
import argparse
from dquality.check import hdf as hdf_check
from os.path import expanduser

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("instrument", help="instrument name, name should have a␣
↪matching directory in the .dquality folder")
    parser.add_argument("fname", help="file name to do the tag dependencies checks on
↪")

    args = parser.parse_args()
    instrument = args.instrument
    fname = args.fname

    home = expanduser("~")
    conf = os.path.join(home, ".dquality", instrument)
```

```
66
67      bad_indexes = hdf_check(conf, fname)
68      return bad_indexes
69
70
71  if __name__ == "__main__":
72      main(sys.argv[1:])
```

Dependency Check

DQuality Data Dependency check example. (Download file: `hdf_dependency_check.py`)

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3
4   # ######################################################################
5   # Copyright (c) 2015, UChicago Argonne, LLC. All rights reserved.      #
6   #                                                                      #
7   # Copyright 2015. UChicago Argonne, LLC. This software was produced    #
8   # under U.S. Government contract DE-AC02-06CH11357 for Argonne National #
9   # Laboratory (ANL), which is operated by UChicago Argonne, LLC for the #
10  # U.S. Department of Energy. The U.S. Government has rights to use,    #
11  # reproduce, and distribute this software.  NEITHER THE GOVERNMENT NOR #
12  # UChicago Argonne, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR     #
13  # ASSUMES ANY LIABILITY FOR THE USE OF THIS SOFTWARE.  If software is  #
14  # modified to produce derivative works, such modified software should  #
15  # be clearly marked, so as not to confuse it with the version available #
16  # from ANL.                                                            #
17  #                                                                      #
18  # Additionally, redistribution and use in source and binary forms, with #
19  # or without modification, are permitted provided that the following   #
20  # conditions are met:                                                  #
21  #                                                                      #
22  #     * Redistributions of source code must retain the above copyright #
23  #       notice, this list of conditions and the following disclaimer.  #
24  #                                                                      #
25  #     * Redistributions in binary form must reproduce the above copyright #
26  #       notice, this list of conditions and the following disclaimer in #
27  #       the documentation and/or other materials provided with the     #
28  #       distribution.                                                  #
29  #                                                                      #
30  #     * Neither the name of UChicago Argonne, LLC, Argonne National    #
31  #       Laboratory, ANL, the U.S. Government, nor the names of its     #
32  #       contributors may be used to endorse or promote products derived #
33  #       from this software without specific prior written permission.  #
34  #                                                                      #
35  # THIS SOFTWARE IS PROVIDED BY UChicago Argonne, LLC AND CONTRIBUTORS  #
36  # "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT    #
37  # LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS    #
38  # FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UChicago  #
39  # Argonne, LLC OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,     #
40  # INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, #
41  # BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;     #
42  # LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER     #
43  # CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT   #
```

```
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN       #
# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE         #
# POSSIBILITY OF SUCH DAMAGE.                                             #
# ########################################################################
"""
Please make sure the installation :ref:`pre-requisite-reference-label` are met.

This script is specific for beamline 32id.

This example takes two mandatory parameters:
instrument: a string defining the detector that will be used. User can enter one of these choices:
'nanotomo', 'microtomo'.
The instrument determines a configuration file that will be used.
file: a file to be verified for dependencies according to schema.

This script calls dependency_check verifier.

"""
import sys
import os
import argparse
from dquality.check import hdf_dependency as dependency_check
from os.path import expanduser

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("instrument", help="instrument name, name should have a matching directory in the .dquality folder")
    parser.add_argument("fname", help="file name to do the tag dependencies checks on")

    args = parser.parse_args()
    instrument = args.instrument
    fname = args.fname

    home = expanduser("~")
    conf = os.path.join(home, ".dquality", instrument)

    bad_indexes = dependency_check(conf, fname)
    return bad_indexes


if __name__ == "__main__":
    main(sys.argv[1:])
```

File Monitor

DQuality Folder Monitoring example. (Download file: `monitor_check.py`)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ########################################################################
```

```
# Copyright (c) 2015, UChicago Argonne, LLC. All rights reserved.          #
#                                                                          #
# Copyright 2015. UChicago Argonne, LLC. This software was produced        #
# under U.S. Government contract DE-AC02-06CH11357 for Argonne National    #
# Laboratory (ANL), which is operated by UChicago Argonne, LLC for the     #
# U.S. Department of Energy. The U.S. Government has rights to use,        #
# reproduce, and distribute this software.  NEITHER THE GOVERNMENT NOR     #
# UChicago Argonne, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR         #
# ASSUMES ANY LIABILITY FOR THE USE OF THIS SOFTWARE.  If software is      #
# modified to produce derivative works, such modified software should      #
# be clearly marked, so as not to confuse it with the version available    #
# from ANL.                                                                #
#                                                                          #
# Additionally, redistribution and use in source and binary forms, with    #
# or without modification, are permitted provided that the following       #
# conditions are met:                                                      #
#                                                                          #
#     * Redistributions of source code must retain the above copyright     #
#       notice, this list of conditions and the following disclaimer.      #
#                                                                          #
#     * Redistributions in binary form must reproduce the above copyright  #
#       notice, this list of conditions and the following disclaimer in    #
#       the documentation and/or other materials provided with the         #
#       distribution.                                                      #
#                                                                          #
#     * Neither the name of UChicago Argonne, LLC, Argonne National        #
#       Laboratory, ANL, the U.S. Government, nor the names of its         #
#       contributors may be used to endorse or promote products derived    #
#       from this software without specific prior written permission.      #
#                                                                          #
# THIS SOFTWARE IS PROVIDED BY UChicago Argonne, LLC AND CONTRIBUTORS      #
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT        #
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS        #
# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UChicago      #
# Argonne, LLC OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,         #
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,     #
# BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;         #
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER         #
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT       #
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN        #
# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE          #
# POSSIBILITY OF SUCH DAMAGE.                                              #
# ##########################################################################
import sys
import os
import argparse
from dquality.check import monitor as monitor_check
from os.path import expanduser

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("instrument", help="instrument name, name should have a␣
↪matching directory in the .dquality folder")
    parser.add_argument("fname", help="folder name to monitor for files")
    parser.add_argument("numfiles", help="number of files to monitor for")

    args = parser.parse_args()
```

```
    instrument = args.instrument
    fname = args.fname
    num_files = args.numfiles

    home = expanduser("~")
    conf = os.path.join(home, ".dquality", instrument)

    bad_indexes = monitor_check(conf, fname, num_files)
    return bad_indexes


if __name__ == "__main__":
    main(sys.argv[1:])
```

PV Check

DQuality PV check example. (Download file: `pv_check.py`)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ##########################################################################
# Copyright (c) 2015, UChicago Argonne, LLC. All rights reserved.          #
#                                                                          #
# Copyright 2015. UChicago Argonne, LLC. This software was produced        #
# under U.S. Government contract DE-AC02-06CH11357 for Argonne National    #
# Laboratory (ANL), which is operated by UChicago Argonne, LLC for the     #
# U.S. Department of Energy. The U.S. Government has rights to use,        #
# reproduce, and distribute this software.  NEITHER THE GOVERNMENT NOR     #
# UChicago Argonne, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR         #
# ASSUMES ANY LIABILITY FOR THE USE OF THIS SOFTWARE.  If software is      #
# modified to produce derivative works, such modified software should      #
# be clearly marked, so as not to confuse it with the version available    #
# from ANL.                                                                #
#                                                                          #
# Additionally, redistribution and use in source and binary forms, with    #
# or without modification, are permitted provided that the following       #
# conditions are met:                                                      #
#                                                                          #
#     * Redistributions of source code must retain the above copyright     #
#       notice, this list of conditions and the following disclaimer.      #
#                                                                          #
#     * Redistributions in binary form must reproduce the above copyright  #
#       notice, this list of conditions and the following disclaimer in    #
#       the documentation and/or other materials provided with the         #
#       distribution.                                                      #
#                                                                          #
#     * Neither the name of UChicago Argonne, LLC, Argonne National        #
#       Laboratory, ANL, the U.S. Government, nor the names of its         #
#       contributors may be used to endorse or promote products derived    #
#       from this software without specific prior written permission.      #
#                                                                          #
# THIS SOFTWARE IS PROVIDED BY UChicago Argonne, LLC AND CONTRIBUTORS      #
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT        #
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS        #
```

```
# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UChicago      #
# Argonne, LLC OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,         #
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,     #
# BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;         #
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER         #
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT       #
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN        #
# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE          #
# POSSIBILITY OF SUCH DAMAGE.                                              #
# ##########################################################################
"""
Please make sure the installation :ref:`pre-requisite-reference-label` are met.

This script is specific for beamline 32id.

This example takes two mandatory parameters:
instrument: a string defining the detector that will be used. User can enter one of␣
↪these choices:
'nanotomo', 'microtomo'.
The instrument determines a configuration file that will be used.
file: a file to be verified for dependencies according to schema.

This script calls hdf_check verifier.

"""
import sys
import os
import argparse
from dquality.check import pv as pv_check
from os.path import expanduser

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("instrument", help="instrument name, name should have a␣
↪matching directory in the .dquality folder")

    args = parser.parse_args()
    instrument = args.instrument

    home = expanduser("~")
    conf = os.path.join(home, ".dquality", instrument)

    bad_indexes = pv_check(conf)
    return bad_indexes


if __name__ == "__main__":
    main(sys.argv[1:])
```

Realtime Check

DQuality real time check example. (Download file: `realtime_check.py`)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# ######################################################################
# Copyright (c) 2015, UChicago Argonne, LLC. All rights reserved.      #
#                                                                      #
# Copyright 2015. UChicago Argonne, LLC. This software was produced    #
# under U.S. Government contract DE-AC02-06CH11357 for Argonne National   #
# Laboratory (ANL), which is operated by UChicago Argonne, LLC for the    #
# U.S. Department of Energy. The U.S. Government has rights to use,    #
# reproduce, and distribute this software.  NEITHER THE GOVERNMENT NOR    #
# UChicago Argonne, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR     #
# ASSUMES ANY LIABILITY FOR THE USE OF THIS SOFTWARE.  If software is     #
# modified to produce derivative works, such modified software should     #
# be clearly marked, so as not to confuse it with the version available   #
# from ANL.                                                            #
#                                                                      #
# Additionally, redistribution and use in source and binary forms, with   #
# or without modification, are permitted provided that the following      #
# conditions are met:                                                  #
#                                                                      #
#     * Redistributions of source code must retain the above copyright    #
#       notice, this list of conditions and the following disclaimer.     #
#                                                                      #
#     * Redistributions in binary form must reproduce the above copyright #
#       notice, this list of conditions and the following disclaimer in   #
#       the documentation and/or other materials provided with the        #
#       distribution.                                                  #
#                                                                      #
#     * Neither the name of UChicago Argonne, LLC, Argonne National       #
#       Laboratory, ANL, the U.S. Government, nor the names of its        #
#       contributors may be used to endorse or promote products derived   #
#       from this software without specific prior written permission.     #
#                                                                      #
# THIS SOFTWARE IS PROVIDED BY UChicago Argonne, LLC AND CONTRIBUTORS     #
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT       #
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS       #
# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL UChicago     #
# Argonne, LLC OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,        #
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,    #
# BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;        #
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER        #
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT      #
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN       #
# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE         #
# POSSIBILITY OF SUCH DAMAGE.                                          #
# ######################################################################
"""
Please make sure the installation :ref:`pre-requisite-reference-label` are met.

This script is specific for beamline 32id.

This example takes one mandatory parameter, and three optional:
instrument: a string defining the detector that will be used. User can enter one of␣
↪these choices:
'32id_nano', '32id_micro'. The instrument determines the directory to look for a␣
↪configuration file that will be used.
type: optional parameter, data type to be verified (i.e. data_dark, data_white or␣
↪data), defaulted to 'data'
```

```
report_file: optional parameter, name of report file, defaulted to None
report_type: optional parameter, report type, currently supporting REPORT_NONE, REPORT_ERRORS, and REPORT_FULL,
defaulted to REPORT_FULL.

This script calls real_time verifier.

"""
import sys
import os
import argparse
import dquality.check_rt as rt
from os.path import expanduser

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("instrument", help="instrument name, name should have a matching directory in the .dquality folder")
    parser.add_argument("--report_file", default=None, help="optional, name of report file")
    parser.add_argument("--sequence", default=None, help="optional, expected sequence of data types")

    args = parser.parse_args()
    instrument = args.instrument
    report_file = args.report_file
    sequence = args.sequence

    home = expanduser("~")
    conf = os.path.join(home, ".dquality", instrument)

    bad_indexes = rt.realtime(conf, report_file, sequence)
    return bad_indexes


if __name__ == "__main__":
    main(sys.argv[1:])

# sequence example
# variableDict = {'PreDarkImages': 5,
#              'PreWhiteImages': 10,
#              'Projections': 60,
#              'PostDarkImages': 5,
#              'PostWhiteImages': 10}
# sequence = []
# index = -1
# try:
#     images = variableDict['PreDarkImages']
#     index += images
#     sequence.append(('data_dark', index))
# except KeyError:
#     pass
# try:
#     images = variableDict['PreWhiteImages']
#     index += images
#     sequence.append(('data_white', index))
# except KeyError:
```

```
#     pass
# try:
#     images = variableDict['Projections']
#     index += images
#     sequence.append(('data', index))
# except KeyError:
#     pass
# try:
#     images = variableDict['PostDarkImages']
#     index += images
#     sequence.append(('data_dark', index))
# except KeyError:
#     pass
# try:
#     images = variableDict['PostWhiteImages']
#     index += images
#     sequence.append(('data_white', index))
# except KeyError:
#     pass
#
# json_sequence = json.dumps(sequence).replace(" ","")
#
```

CHAPTER 5

Frequently asked questions

Here's a list of questions.

Questions

How can I report bugs?

The easiest way to report bugs or get help is to open an issue on GitHub. Simply go to the project GitHub page, click on Issues in the right menu tab and submit your report or question.

CHAPTER 6

Credits

References

CHAPTER 7

Howto

This file provides instructions how to use the framework to add more features.

How to define quality check functions in quality_checks.json file.

This framework classifies two types of quality checks: the frame verification check, and statistical check. Frame verification check calculates characteristic of a given frame (for example mean value) and tests the result against limits. Statistical quality check uses previously stored results of frame verification check to evaluate the current result.

Follow the steps below to add a frame verification check: - add the method in dquality.common.qualitychecks.py file. The signature should be: function_name(data, index, results, all_limits). Look at the example method "validate_mean_signal_intensity" for the meaning of the parameters. - define quality id in a dquality/common/constants.py file that is unique in respect to other quality checks constants, and update mapper in the constants.py file. The constant value should be less than 100. - add a dictionary entry to the quality_checks.json file where key is the string representation of a constant defining the method, and value is an empty list.

Follow the steps below to add a statistical quality check:

- add the method to dquality/common/qualitychecks.py file. The signature should be: function_name(result, aggregate, results, all_limits). Look at the example statistical method "validate_stat_mean" for the meaning of the parameters.
- define quality id in a dquality/common/constants.py file that is unique in respect to other quality checks constants, and update mapper in the constants.py file. The constant value should be greater or equal 100.
- find a key of the frame verification check that the statistical check uses, and add an entry to the corresponding list, where the entry is a a constant defining the statistical check method.

example:

The following quality_checks.json file { "QUALITYCHECK_MEAN" : ["STAT_MEAN"], "QUALITYCHECK_STD" : []} defines that each frame will be verified by frame verification functions: defined by constants: QUALITYCHECK_MEAN, and QUALITYCHECK_STD. Right aftre a frame is evaluated with QUALITYCHECK_MEAN, the statistical quality check STAT_MEAN evaluates the frame further, using the previous results.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Bibliography

[C1] Data Quality Control. http://cbs.fas.harvard.edu/science/core-facilities/neuroimaging/information-investigators/qc. Accessed: 2016-03-02.

[C2] MRI Quality Control. http://cbs.fas.harvard.edu/usr/mcmains/CBS_MRI_Quality_Control_Workshop.pdf. Accessed: 2016-03-02.

[C3] L. W. Goldman. Principles of CT: radiation dose and image quality. *J Nucl Med Technol*, 35(4):213–225, Dec 2007.

[C4] Pedram Mohammadi, Abbas Ebrahimi-Moghadam, and Shahram Shirani. Subjective and objective quality assessment of image: A survey. *CoRR*, 2014. URL: http://arxiv.org/abs/1406.7799.

Python Module Index

d

Index