Mike Rumore

AbboTT Labs  Lake Forest, IL

R/SQL Architecture

Table of Contents

[1 Apollo Requirements 4](#_Toc23150558)

[1.1 General Requirements 4](#_Toc23150559)

[1.2 input.csv 4](#_Toc23150560)

[1.3 config.csv 5](#_Toc23150561)

[1.3.1 DX via SAML Connection 5](#_Toc23150562)

[1.3.2 DX via Custom Connection 6](#_Toc23150563)

[1.3.3 Reliability via DSN Connection 6](#_Toc23150564)

[1.3.4 Reliability via NON-DSN Connection 6](#_Toc23150565)

[1.3.5 DX via Spark using YARN 6](#_Toc23150566)

[1.3.6 DX via SPARK using Apache Livy 7](#_Toc23150567)

[1.3.7 IDA 7](#_Toc23150568)

[1.4 results.csv 7](#_Toc23150569)

[1.5 chart\_data.csv 8](#_Toc23150570)

[1.6 errors.csv 10](#_Toc23150571)

[1.7 Output Files and Returned Status to Apollo 10](#_Toc23150572)

[2 R/SQL Script Architecture 10](#_Toc23150573)

[2.1 R/SQL Flat Script Layout 10](#_Toc23150574)

[2.2 common\_utils.R 11](#_Toc23150575)

[2.2.1 reliability\_dsn\_connect\_to\_db 11](#_Toc23150576)

[2.2.2 reliability\_non\_dsn\_connect\_to\_db 11](#_Toc23150577)

[2.2.3 reliability\_connect\_to\_db 11](#_Toc23150578)

[2.2.4 dx\_custom\_connect\_to\_db 11](#_Toc23150579)

[2.2.5 dx\_saml\_connect\_to\_db 11](#_Toc23150580)

[2.2.6 dx\_connect\_to\_db 11](#_Toc23150581)

[2.2.7 spark\_connect\_to\_db 11](#_Toc23150582)

[2.2.8 connect\_to\_db 11](#_Toc23150583)

[2.2.9 disconnect\_from\_db 11](#_Toc23150584)

[2.2.10 read\_test\_dates 12](#_Toc23150585)

[2.2.11 get\_test\_period 12](#_Toc23150586)

[2.2.12 read\_csv\_file 12](#_Toc23150587)

[2.2.13 query\_subs 12](#_Toc23150588)

[2.2.14 make\_save\_to\_file 12](#_Toc23150589)

[2.2.15 save\_to\_file 12](#_Toc23150590)

[2.2.16 make\_error\_handler 13](#_Toc23150591)

[2.2.17 handler 13](#_Toc23150592)

[2.2.18 phm\_patterns\_sk 13](#_Toc23150593)

[2.2.19 occurred 13](#_Toc23150594)

[2.2.20 total\_errors 13](#_Toc23150595)

[2.2.21 file\_name 13](#_Toc23150596)

[2.2.22 make\_write\_results 13](#_Toc23150597)

[2.2.23 write\_results 13](#_Toc23150598)

[2.2.24 make\_write\_data 13](#_Toc23150599)

[2.2.25 write\_data 13](#_Toc23150600)

[2.2.26 empty\_results 14](#_Toc23150601)

[2.2.27 exec\_query 14](#_Toc23150602)

[2.2.28 default\_post\_flagged\_processing 14](#_Toc23150603)

[2.2.29 default\_pre\_flagged\_processing 14](#_Toc23150604)

[2.2.30 default\_spark\_load\_data 14](#_Toc23150605)

[2.2.31 default\_generate\_suppression 14](#_Toc23150606)

[2.3 main.R 15](#_Toc23150607)

[2.4 algorithm.R 15](#_Toc23150608)

[2.5 R/SQL Ad Hoc Script Layout 16](#_Toc23150609)

[2.6 adhoc\_common\_utils.R 16](#_Toc23150610)

[2.6.1 usage 16](#_Toc23150611)

[2.6.2 read\_test\_dates 16](#_Toc23150612)

[2.6.3 get\_test\_period 16](#_Toc23150613)

[2.6.4 read\_csv\_file 17](#_Toc23150614)

[2.6.5 query\_subs 17](#_Toc23150615)

[2.6.6 make\_save\_to\_file 17](#_Toc23150616)

[2.6.7 save\_to\_file 17](#_Toc23150617)

[2.6.8 make\_error\_handler 17](#_Toc23150618)

[2.6.9 handler 17](#_Toc23150619)

[2.6.10 phm\_patterns\_sk 17](#_Toc23150620)

[2.6.11 occurred 18](#_Toc23150621)

[2.6.12 total\_errors 18](#_Toc23150622)

[2.6.13 file\_name 18](#_Toc23150623)

[2.6.14 make\_write\_results 18](#_Toc23150624)

[2.6.15 write\_results 18](#_Toc23150625)

[2.6.16 make\_write\_data 18](#_Toc23150626)

[2.6.17 write\_data 18](#_Toc23150627)

[2.6.18 empty\_results 18](#_Toc23150628)

[2.6.19 reliability\_dsn\_connect\_to\_db 18](#_Toc23150629)

[2.6.20 reliability\_non\_dsn\_connect\_to\_db 19](#_Toc23150630)

[2.6.21 reliability\_connect\_to\_db 19](#_Toc23150631)

[2.6.22 dx\_custom\_connect\_to\_db 19](#_Toc23150632)

[2.6.23 dx\_saml\_connect\_to\_db 19](#_Toc23150633)

[2.6.24 dx\_linux\_saml\_connect\_to\_db 19](#_Toc23150634)

[2.6.25 dx\_connect\_to\_db 19](#_Toc23150635)

[2.6.26 ida\_connect\_to\_db 19](#_Toc23150636)

[2.6.27 spark\_connect\_to\_db 19](#_Toc23150637)

[2.6.28 connect\_to\_db 19](#_Toc23150638)

[2.6.29 disconnect\_from\_db 20](#_Toc23150639)

[2.6.30 exec\_query 20](#_Toc23150640)

[2.6.31 default\_post\_flagged\_processing 20](#_Toc23150641)

[2.6.32 process\_results 20](#_Toc23150642)

[2.6.33 run\_algorithm 20](#_Toc23150643)

[2.6.34 default\_spark\_load\_data 21](#_Toc23150644)

[2.6.35 default\_generate\_suppression 21](#_Toc23150645)

[2.6.36 run\_group\_algorithm 21](#_Toc23150646)

[2.6.37 real\_main 21](#_Toc23150647)

[2.6.38 main 21](#_Toc23150648)

[2.7 main.R 22](#_Toc23150649)

[2.8 RLIB 22](#_Toc23150650)

[2.9 Testing 22](#_Toc23150651)

[2.9.1 Testing with test\_dates.csv File 23](#_Toc23150652)

[2.9.2 Testing So Only Flagged-Query Is Executed 24](#_Toc23150653)

[2.9.3 Query.sql File 24](#_Toc23150654)

[3 Algorithms 24](#_Toc23150655)

[3.1 Current Algorithms 24](#_Toc23150656)

[3.2 Mapping input.csv Parameters to Variables in Query Templates 25](#_Toc23150657)

[3.3 Charting 27](#_Toc23150658)

[3.3.1 Charting in results.csv 27](#_Toc23150659)

[3.3.2 Charting in chart\_data.csv 27](#_Toc23150660)

[3.4 Adding a New Algorithm 27](#_Toc23150661)

[3.4.1 Customizing Algorithms 28](#_Toc23150662)

# Apollo Requirements

## General Requirements

Following are the requirements for R algorithms to be executed against BDAA/Dx data source using Athena

* Algorithm client shall create the input parameter (input.csv) file and make it available to R algorithm
* Athena and on-premise Reliability database connections would be written in a separate R script and the algorithms could source the connection
* script. R algorithms should not be hard-coding the connection strings/database credentials as part of algorithm logic/code.
* Algorithms shall loop thru all the algorithms defined using the group routine, process the data and output results for each algorithm.
* Algorithm shall output both flagged and healthy instrument data for each algorithm.
* Get available instruments in the last 24 hours, from all the logs that the algorithm is querying and identify healthy/flagged conditions for that data set.
* Algorithm execution results shall be written to results.csv file
* Algorithm execution errors shall be written to errors.csv file

Additional requirements/assumptions:

* Execution timeouts need to be defined for the algorithms? (not implemented)
* Additional libraries required for the Algorithm(s): Checkpoint package install option shall be used to install the required packages, as needed

## input.csv

The input.csv file format is shown next. This file is created by Apollo.

| **Column Name** | **Column Type** | **Comments** |
| --- | --- | --- |
| ALGORITHM\_NAME | VARCHAR(200) | Algorithm Name, as entered in the algorithm definition |
| PHM\_PATTERNS\_SK | NUMBER | PHM\_PATTERNS\_SK from PHM\_PATTERNS table |
| PARAMETER\_NAME | VARCHAR(500) | Algorithm Parameter names, as configured in the algorithm definition |
| PARAMETER\_VALUE | VARCHAR(4000) | Algorithm Parameters values, as configured in the algorithm definition |

## config.csv

The config.csv file format is shown next. This file is created by Apollo. There are several different formats depending on the methods used to connect to the databases. Examples are given for each database connection type. A config.csv file can have more than one set of parameters since the simultaneous connections to DX via Athena or via Spark, and to the Reliability database is common.

| **Column Name** | **Column Type** | **Comments** |
| --- | --- | --- |
| NAME | VARCHAR | Name of parameter used in R script |
| VALUE | VARCHAR | Value of parameter used in R |

For development test, the config.csv file must be created by hand. This process will be described in a later section on testing.

### DX via SAML Connection

| **Name** | **Value** |
| --- | --- |
| ATHENA\_JDBC\_DRIVER\_CLASS | com.simba.athena.jdbc.Driver |
| ATHENA\_JDBC\_CLASSPATH | C:/Users/rumormx/Documents/sandbox/phm/jlib/athena/AthenaJDBC41\_2.0.7.jar |
| ATHENA\_LOGLEVEL | 6 |
| ATHENA\_LOGPATH | logs |
| ATHENA\_WORKGROUP | add\_service\_dx\_readonly |
| ATHENA\_USERESULTSETSTREAMING | 0 |
| ATHENA\_S3OUTPUTLOCATION | s3://abt-bdaa-test-us-east-1-sandbox/athena/RUMORMX |
| ATHENA\_AWSCREDENTIALSPROVIDERCLASS | com.simba.athena.amazonaws.auth.profile.ProfileCredentialsProvider |
| ATHENA\_AWSCREDENTIALSPROVIDERARGUMENTS | saml |
| ATHENA\_S3\_STAGING\_DIR | s3://abt-bdaa-test-us-east-1-sandbox/athenaquerylog |
| ATHENA\_DB\_CONN\_STRING | jdbc:awsathena://awsregion=us-east-1 |
| ATHENA\_CREDENTIAL\_PROVIDER | saml |
| ATHENA\_DB\_USER | XXXXXXXX |
| ATHENA\_DB\_PASSWORD | YYYYYYYY |

### DX via Custom Connection

| **Name** | **Value** |
| --- | --- |
| ATHENA\_JDBC\_DRIVER\_CLASS | com.simba.athena.jdbc.Driver |
| ATHENA\_JDBC\_CLASSPATH | C:/Users/rumormx/Documents/sandbox/phm/jlib/athena/AthenaJDBC41\_2.0.7.jar |
| ATHENA\_LOGLEVEL | 6 |
| ATHENA\_LOGPATH | logs |
| ATHENA\_WORKGROUP | add\_service\_dx\_readonly |
| ATHENA\_USERESULTSETSTREAMING | 0 |
| ATHENA\_S3OUTPUTLOCATION | s3://abt-bdaa-test-us-east-1-sandbox/athena/RUMORMX |
| ATHENA\_AWSCREDENTIALSPROVIDERCLASS | com.simba.athena.amazonaws.auth.profile.ProfileCredentialsProvider |
| ATHENA\_AWSCREDENTIALSPROVIDERARGUMENTS | saml |
| ATHENA\_S3\_STAGING\_DIR | s3://abt-bdaa-test-us-east-1-sandbox/athenaquerylog |
| ATHENA\_DB\_CONN\_STRING | jdbc:awsathena://awsregion=us-east-1 |
| ATHENA\_CREDENTIAL\_PROVIDER | custom |
| ATHENA\_DB\_USER | XXXXXXXX |
| ATHENA\_DB\_PASSWORD | YYYYYYYY |

### Reliability via DSN Connection

| **Name** | **Value** |
| --- | --- |
| RELIABILITY\_DB\_SERVER | WQ00418P\SQLP01 |
| RELIABILITY\_DB\_NAME | Reliability |
| RELIABILITY\_DB\_USER | gsr\_reader |
| RELIABILITY\_DB\_PASSWORD | Rel11APP |
| RELIABILITY\_DB\_CONN\_TYPE | DSN |

### Reliability via NON-DSN Connection

| **Name** | **Value** |
| --- | --- |
| SQL\_SERVER\_ODBC\_DRIVER\_NAME | SQL Server |
| RELIABILITY\_DB\_SERVER\_NAME | WQ00418P\SQLP01 |
| RELIABILITY\_DB\_USER | Reliability |
| RELIABILITY\_DB\_PASSWORD | Rel11APP |
| RELIABILITY\_DB\_INSTANCE\_PORT | 63430 |
| RELIABILITY\_DB\_PORT | 1433 |
| RELIABILITY\_DB\_CONN\_TYPE | NON-DSN |

### DX via Spark using YARN

| **Name** | **Value** |
| --- | --- |
| SPARK\_HOME | /usr/lib/spark |
| HADOOP\_CONF\_DIR | /etc/hadoop/conf |
| YARN\_CONF\_DIR | /etc/hadoop/conf |
| SPARK\_EXECUTOR\_INSTANCES | 4 |
| SPARK\_EXECUTOR\_CORES | 4 |
| SPARK\_EXECUTOR\_MEMORY | 20GB |
| SPARK\_DYANMICALLOCATION\_ENABLED | false |
| SPARK\_MASTER | yarn-client |
| SPARK\_VERSION | 2.3.1 |

### DX via SPARK using Apache Livy

| **Name** | **Value** |
| --- | --- |
| SPARK\_HOME | /usr/lib/spark |
| HADOOP\_CONF\_DIR | /etc/hadoop/conf |
| YARN\_CONF\_DIR | /etc/hadoop/conf |
| SPARK\_EXECUTOR\_INSTANCES | 4 |
| SPARK\_EXECUTOR\_CORES | 4 |
| SPARK\_EXECUTOR\_MEMORY | 20GB |
| SPARK\_DYANMICALLOCATION\_ENABLED | false |
| LIVY\_MASTER | TBD |

### IDA

| **Name** | **Value** |
| --- | --- |
| IDA\_DB\_NAME | pabbto |
| IDA\_DB\_HOST | ux00147p-scan.oneabbott.com |
| IDA\_DB\_PORT | 1521 |
| IDA\_DB\_USER | SVC\_PHM\_CONNECT |
| IDA\_DB\_PASSWORD | svc\_phmc\_0771 |
| IDA\_JDBC\_CLASSPATH | ,C:/Users/rumormx/Documents/sandbox/phm/jlib/OJDBC-Full/ojdbc6.jar |
| IDA\_JDBC\_DSN\_FORMAT | jdbc:oracle:thin:@//%s:%s/%s |
| IDA\_JDBC\_DRIVER\_CLASS | oracle.jdbc.OracleDriver |

## results.csv

The results.csv file is created by the algorithm. The file format is shown next.

| **Name** | **Required?** | **Type** | **Comments** |
| --- | --- | --- | --- |
| PHM\_PATTERNS\_SK | Yes | NUMBER | Algorithm unique ID |
| PL | Yes | VARCHAR(5) | Product Line Code (e.g., 115, 116, 117, 205, 210, 214, etc.) |
| SN | Yes | VARCHAR(20) | Instrument Serial Number |
| FLAG\_DATE | Yes | VARCHAR(20) | Instrument local Date/Time of when this instrument has been flagged. Value should be in YYYYMMDDHH24MISS format  If instrument is marked as flagged (FLAG\_YN =1)  -> Flag Date - would be the max of instrument local date/time over a period of time that the algorithm ran against ->  format: YYYYMMDDHH24MISS  If instrument is marked as healthy (FLAG\_YN=0)  -> Flag Date -> would be the current day in YYYYMMDD format |
| CHART\_DATA\_VALUE | No | NUMBER(15,5) | Counter value. Required, If FLAG\_YN is 1. Make sure to set DEVICE\_VALUE to >= 1 if FLAG\_YN = 1. Maps to DEVICE\_VALUE in  PHM\_ALG\_OUTPUT table |
| FLAG\_YN | Yes | NUMBER(1) | Flag to identify if a instrument is healthy or unhealthy. Flagged -> 1; Healthy -> 0 |
| IHN\_LEVEL3\_DESC | No | VARCHAR(150) | 1) Algorithm has single experience code assigned for a PL.  -> If the results.csv file populate IHN\_LEVEL3\_DESC value then R processor takes it and persists as IHN\_LEVEL3\_DECSC in  PHM\_ALG\_OUTPUT table  -> If the results.csv file does not populate IHN\_LEVEL3\_DESC then R processor picks up what is assigned to IHN\_LEVEL3\_DESC  parameter in the algorithm definition and populates it in PHM\_ALG\_OUTPUT table  2) Algorithm has multiple experience codes assigned for a single PL  -> In this case, results.csv should always populate IHN\_LEVEL3\_DESC and it should also match with LEVEL 3 description assigned in  TCT. System will compare the TCT LEVEL 3 with IHN\_LEVEL3\_DESC column and pick up the right experience code.  -> If the R output (results.csv) does not populate IHN\_LEVEL3\_DESC value then leave this field as blank in PHM\_ALG\_OUTPUT table. |

## chart\_data.csv

| **Name** | **Required?** | **Type** | **Comments** |
| --- | --- | --- | --- |
| PHM\_PATTERNS\_SK | Yes | NUMBER | PHM\_PATTERNS\_SK (algorithm unique key), represents the algorithm |
| PL | Yes | VARCHAR2(5) | Product Line Code (Ex: 115, 116, 117, 205, 210, 214 etc.,) |
| SN | Yes | VARCHAR2(30) | Instrument Serial Number |
| FLAG\_DATE | Yes | VARCHAR2(20) | Instrument local Date/Time of when this instrument has been flagged. Value should be in YYYYMMDDHH24MISS format  If instrument is marked as flagged (FLAG\_YN =1)  -> Flag Date - would be the max of instrument local date/time over a period of time that the algorithm ran against -> format: YYYYMMDDHH24MISS If instrument is marked as healthy (FLAG\_YN=0) -> Flag Date -> would be the current day in YYYYMMDD format |
| CHART\_DATA\_VALUE | YES | NUMBER(15,5) | Counter value. Required, If FLAG\_YN is 1. Make sure to set DEVICE\_VALUE to >= 1 if FLAG\_YN = 1. Maps to DEVICE\_VALUE in PHM\_ALG\_OUTPUT table  Note: May not required to be populated by the algorithms. Deborah to confirm. |
| DATA\_SERIES | YES | VARCHAR2(5) | Data series value  For v5.0 scope, only WAM algorithms shall output chart\_data.csv file. DATA\_SERIES field will have **hourly** data points for WZ\_PROBE information (Ex: 1.2, 1.3, 2.1, 2.2) |

## errors.csv

| **Name** | **Required?** | **Type** | **Comments** |
| --- | --- | --- | --- |
| PHM\_PATTERNS\_SK | Yes | NUMBER | PHM\_PATTERNS\_SK (algorithm unique key), represents the algorithm |
| ERROR\_MESSAGE | Yes | VARCHAR2(4000) | Error message |

## Output Files and Returned Status to Apollo

The following table summarizes the various exit conditions and what must happen when the algorithm exits.

| **Conditions** | **R-script exit status** | **Errors.csv** | **Results.csv** |
| --- | --- | --- | --- |
| No processing errors and instruments are flagged. | 0 | Not created | Results.csv file contains flagged instruments and healthy instruments if MODULESN QUERY TEMPLATE is defined. |
| No processing errors occurred and no instruments are flagged. | 0 | Not created | Results.csv is created and it will only contains healthy instruments if the MODULESSN QUERY TEMPLATE is defined. |
| Processing errors at query execution level. Data are generated. | -1 | Created containing any error message issued by R or any stop() message fired by the algorithm | Results.csv may exist if any data were processed. |
| Processing errors before database initialization. | -1 | Created and contains the error message. | Not created. |
| Processing errors after queries are processed. | -1 | Created and contains all error messages. | Results.csv may be created. |

# R/SQL Script Architecture

## R/SQL Flat Script Layout

This script format consists of three files: common\_utils.R, main.R and algorithm.R. The files main.R and common\_utils.R are shared by several algorithms and are the same for each algorithm. The file common\_utils.R contains R routines which are used in all main.R and algorithm.R files. Algorithm.R contains the SQL queries, function overrides and constants which define the algorithm. The following section will describe what each of these files contains or should contain for the case of algorithm.R.

## common\_utils.R

Common utilities contains functions which are used by all algorithms based on the flat script layout. The functions are described next.

### reliability\_dsn\_connect\_to\_db

Description: connect to Reliability database using an existing DSN.

Parameters: config - data.frame created from reading config file.

Returns: ODBC database connection

### reliability\_non\_dsn\_connect\_to\_db

Description: connect to Reliability database using using DB server,user, password, DB name defined in config file data.frame

Parameters: config - data.frame created from reading config file.

Returns: ODBC database connection

### reliability\_connect\_to\_db

Description: connects to Reliability database using DSN or non-DSN connection method.

Parameters: config - data.frame created from reading config file.

Returns: ODBC database connection.

### dx\_custom\_connect\_to\_db

Description: connects to DX database using a custom JDBC class.

Parameters: config - data.frame created from reading config file.

Returns: JDBC database connection.

### dx\_saml\_connect\_to\_db

Description: connect DX datanbase using a SAML-based connection.

Parameters: config - data.frame created from reading config file.

Returns: JDBC database connection.

### dx\_connect\_to\_db

Description: main entry point for connecting to DX database.

Parameters: config - data.frame created from reading config file.

Returns: JDBC database connection.

### spark\_connect\_to\_db

Description: connect to DX database using Spark

Parameters: config - data.frame created from reading config file.

Returns: SPARK database connection

### connect\_to\_db

Description: main entry point to connect to a remote database: DX via Athena or DX via Spark.

Parameters: config - data.frame created from reading config file.

Returns: JDBC Athena or SPARK database connection

### disconnect\_from\_db

Description: main entry to disconnect from DX database

Parameters: db\_conn - database connection object

Returns: nothing

### read\_test\_dates

Description: read test dates from a csv file

Parameters: filename - name of test dates file

Returns: test dates data.frame (see below)

### get\_test\_period

Description: generate a test period for algorithm

Parameters: number\_of\_days - number of days for algorithm

modulesn\_number\_of\_days - number of days for healthy instrument list.

Returns: data frame with test dates. see below.

data.frame(NAME=c("START\_DATE",

"END\_DATE",

"MODULESN\_START\_DATE",

"MODULESN\_END\_DATE",

"HEALTHY\_FLAG\_DATE"),

VALUE=c(format(now-number\_of\_days, test\_date\_format),

format(now,test\_date\_format),

format(now-modulesn\_number\_of\_days, test\_date\_format),

format(now, test\_date\_format),

format(now-1, healthy\_date\_format)))

### read\_csv\_file

Description: read a csv file with sanity checks

Parameters: filename - path to CSV file

type\_of\_file - label for type of file used in errors

Returns: data.frame defined by contents of CSV file. Comments are allowed, and strings are not converted to factors.

### query\_subs

Description: substitute parameters for variables in query

Parameters: query\_template - query containing variables with the forma "<name of variable>".

substitutions - data frame used for substitutions contains columns which correspond to names and values. the rows are names after the values in the names column.

value\_column\_name - name of column containing the values.

Returns: query with variables replaced with values from the substitutions data.frame. some variables may not be defined using this substitutions data frame.

### make\_save\_to\_file

Description: create function to save data to a file

Parameters: none

Returns: save\_to\_file function and attached scope

### save\_to\_file

Description: function to save data to a file

Parameters: data - data to save

file\_name - file path to store data

Returns: none

### make\_error\_handler

Description: create a function to write error messages to errors.csv

Parameters: none

Returns: pseudo-object error handler

### handler

Description: callback used by try-catch block in case of an error

Parameters: emsg - error message, usually the message used in the stop() call.

Returns: none

### phm\_patterns\_sk

Description:: set/get function

Parameters: phm\_patterns\_sk - new value or NA

Returns: current value of phm\_patterns\_sk

### occurred

Description: set/get function

Parameters: occurred - new value or NA

Returns: current value of occurred

### total\_errors

Description: set/get function

Parameters: total\_errors - new value or NA

Returns: current value of total\_errors

### file\_name

Description: set/get function

Parameters: file\_name - new value or NA

Returns: current value of file\_name

### make\_write\_results

Description: create a function to write all the records to the results file

Parameters: none

Returns: write\_results function and scope

### write\_results

Description: write data to results.csv file

Parameters: results - data to save

keep - list of fields to keep.

Returns: none

### make\_write\_data

Description: create a function to write data out

Parameters: none

Returns: write\_data fucntion and scope

### write\_data

Description: function to save data to a file

Parameters: data - data to save

file\_name - file path to store data

Returns: none

### empty\_results

Description: generate an empty list

Parameters: none

Returns: data frame for results.csv with correct columns, but with zero rows.

### exec\_query

Description: generate a query and execute it. process results afterwards.

Parameters: params - one parameter set from list of parameter sets in input.csv file.

db\_conn - database connection created by connect\_to\_db()

query\_template - query with variables

test\_period - test period data.frame created by get\_test\_period()

Returns: query results set

### default\_post\_flagged\_processing

Description: default function for processing of flagged query results. it does nothing. overwritten by algorithm if necessary.

Parameters: flagged\_results - query results from flagged query

db\_conn - database connection created by connect\_to\_db()

params - one parameter set from list of parameter sets in input.csv file.

test\_period - test period data.frame created by get\_test\_period()

Returns: flagged results after processing

### default\_pre\_flagged\_processing

Description: default function for pre-processing of flagged query results. it does nothing. overwritten by algorithm if necessary.

Parameters: flagged\_results - query results from flagged query

db\_conn - database connection created by connect\_to\_db()

params - one parameter set from list of parameter sets in input.csv file.

test\_period - test period data.frame created by get\_test\_period()

Returns: flagged results after processing

### default\_spark\_load\_data

Description: load parquet files for spark, if needed. default version does nothing. is overwritten by algorithm when needed.

Parameters: db\_conn - database connection created by connect\_to\_db()

params - one parameter set from list of parameter sets in input.csv file.

test\_period - test period data.frame created by get\_test\_period()

Returns: none

### default\_generate\_suppression

Description: default function to generate list of suppressed module SNs. by default it does nothing. it can be overwritten by an algorithm.

Parameters: params - one parameter set from list of parameter sets n input.csv file.

rel\_db\_conn - Reliability database connection

test\_period - test period data.frame created by get\_test\_period()

Returns: list of suppressed SNs

## main.R

The file main.R contains all the high-level logic which defines an algorithm. The steps it executes are as follows:

1. Set working directory.
2. Load required R packages:
   1. DBI
   2. RJDBC
   3. odbc
   4. dplyr
   5. sparklyr
3. Source the common\_utils.R file.
4. Source the algorithm.R file.
5. Start a try-catch block for error processing.
6. Read in config.csv file which contains data for connecting to external databases. This file is created by Apollo.
7. Read in input.csv file which contains all the sets of parameters required for the algorithm. This file is created by Apollo. Once the CSV file is read, then the data split into sets depending on the parameter PHM\_PATTERNS\_SK.
8. Get start and end dates for algorithm.
9. Open connection to DX database.
10. Load parquet files for Spark (if needed).
11. Connect to Reliability database if we have a reliability query.
12. Loop over all parameters sets and repeat the following:
    1. Substitute parameters into flagged query template and execute the query.
    2. Substitute the parameters into healthy module SN query templateand execute the query.
    3. If the Reliability query template is defined, then substitute the parameters into the Reliability query template and execute the query.
    4. Process the query results. Call post\_flagged\_processing() function. The default version does nothing. Each algorithmwill overwrite as needed.
    5. 5) Using Reliability database data, suppress any module SN which were recently visited.
    6. 6) Remove flagged module SN from healthy module SN list.
    7. Write results to results.csv file.
13. End of parameter sets loop. Close database connections.
14. Exit with status equal to zero if no errors occurred, else exit with a status of -1.

## algorithm.R

The file algorithm.R defines the actual algorithm. It contains the following.

1. flagged\_query\_template - template query defining the set of flagged instruments.
2. modulesn\_query\_template - template query defining the potential set of healthy instruments.
3. reliability\_query\_template - template query determining the set of instruments which were visited in the last 45 days.
4. number\_of\_days - number of days for the algorithm.
5. product\_line\_code - product line code for the instruments in the algorithm. Can be NA if the queries determine the product line code.
6. config\_type - database type. Can be dx, ida, or spark.
7. post\_flagged\_processing - algorthm-specific function called after the flagged query is execufed.
8. spark\_load\_data - algorithm-specific function for loading parquet files into Spark memory.

## R/SQL Ad Hoc Script Layout

This script format consists of two files: adhoc\_common\_utils.R and main.R. The file adhov\_common\_utils.R is shared by a few algorithms and is the same for each algorithm. The file adhoc\_common\_utils.R contains R routines which are used in all main. files. Main.R contains the SQL queries, function overrides and constants which define the algorithm. The following section will describe what each of these files contains or should contain for the case of main.R.

## adhoc\_common\_utils.R

Common utilities contains functions which are used by all algorithms based on the adhoc script layout. The functions are described next.

### usage

Description: standard usage message

Parameters: none

Returns: none

### read\_test\_dates

Description: read test dates from a csv file

Parameters: filename - name of test dates file

Returns: test dates data.frame (see below)

### get\_test\_period

Description: generate a test period for algorithm

Parameters: options - getopts data.frame

number\_of\_days - number of days for algorithm

modulesn\_number\_of\_days - number of days for healthy instrument list.

Returns: data frame with test dates. see below.

data.frame(NAME=c("START\_DATE",

"END\_DATE",

"MODULESN\_START\_DATE",

"MODULESN\_END\_DATE",

"HEALTHY\_FLAG\_DATE"),

VALUE=c(format(now-number\_of\_days,

test\_date\_format),

format(now,

test\_date\_format),

format(now-modulesn\_number\_of\_days,

test\_date\_format),

format(now,

test\_date\_format),

format(now-1,

healthy\_date\_format)))

### read\_csv\_file

Description: read a csv file with sanity checks

Parameters: filename - path to CSV file

type\_of\_file - label for type of file used in errors

Returns: data.frame defined by contents of CSV file. Comments starting with '#' are allowed and strings are not converted

to factors.

### query\_subs

Description: substitute parameters for variables in query

Parameters: query\_template - query containing variables with the format "<name of variable>".

substitutions - data frame used for substitutions contains columns which correspond to names and values. the rows are names after the values in the names column.

value\_column\_name - name of column containing the values.

Returns: query with variables replaced with values from the substitutions data.frame. some variables may not be defined using this substitutions data frame.

### make\_save\_to\_file

Description: create function to save data to a file

Parameters: none

Returns: save\_to\_file function and attached scope

### save\_to\_file

Description: function to save data to a file

Parameters: data - data to save

file\_name - file path to store data

Returns: none

### make\_error\_handler

Description: create a function to write error messages to errors.csv

Parameters: none

Returns: pseudo-object error handler error\_handler <- function()

### handler

Description: callback used by try-catch block in case of an error

Parameters: emsg - error message, usually the message used in the stop() call.

Returns: none

### phm\_patterns\_sk

Description: set/get function

Parameters: phm\_patterns\_sk - new value or NA

Returns: current value of phm\_patterns\_sk

### occurred

Description: set/get function

Parameters: occurred - new value or NA

Returns: current value of occurred

### total\_errors

Description: set/get function

Parameters: total\_errors - new value or NA

Returns: current value of total\_errors

### file\_name

Description: set/get function

Parameters: file\_name - new value or NA

Returns: current value of file\_name

### make\_write\_results

Description: create a function to write all the records to the results file

Parameters: none

Returns: write\_results function and scope

### write\_results

Description: write data to results.csv file

Parameters: options - getopts data.frame

flagged\_records - data to save

keep - list of fields to keep.

Returns: none

### make\_write\_data

Description: create a function to write data out

Parameters: none

Returns: write\_data fucntion and scope

### write\_data

Description: function to save data to a file

Parameters: data - data to save

file\_name - file path to store data

use\_col\_names - write out column names the second and later times the file is accessed.

Returns: none

### empty\_results

Description: generate an empty list

Parameters: none

Returns: data frame for results.csv with correct columns, but with zero rows.

### reliability\_dsn\_connect\_to\_db

Description: connect to Reliability database using an existing DSN.

Parameters: config - data.frame created from reading config file.

Returns: ODBC database connection

### reliability\_non\_dsn\_connect\_to\_db

Description: connect to Reliability database using using DB server, user, password, DB name defined in config file data.frame

Parameters: config - data.frame created from reading config file.

Returns: ODBC database connection

### reliability\_connect\_to\_db

Description: connects to Reliability database using DSN or non-DSN connection method.

Parameters: config - data.frame created from reading config file.

Returns: ODBC database connection

### dx\_custom\_connect\_to\_db

Description: connects to DX database using a custom JDBC class.

Parameters: config - data.frame created from reading config file.

Returns: JDBC database connection

### dx\_saml\_connect\_to\_db

Description: connect DX datanbase using a SAML-based connection.

Parameters: config - data.frame created from reading config file.

Returns: JDBC database connection

### dx\_linux\_saml\_connect\_to\_db

Description: connect DX datanbase using a SAML-based connection.

Parameters: config - data.frame created from reading config file.

Returns: JDBC database connection

### dx\_connect\_to\_db

Description: main entry point for connecting to DX database.

Parameters: config - data.frame created from reading config file.

Returns: JDBC database connection

### ida\_connect\_to\_db

Description: main entry point for connecting to IDA database.

Parameters: config - data.frame created from reading config file.

Returns: JDBC database connection

### spark\_connect\_to\_db

Description: cpnnect to DX database using Spark

Parameters: config - data.frame created from reading config file.

Returns: SPARK database connection

### connect\_to\_db

Description: main entry point to connect to a remote database: DX via Athena or DX via Spark.

Parameters: options - getopts data.frame

Returns: JDBC Athena or SPARK database connection, or IDA JDBC connection

### disconnect\_from\_db

Description: main entry to disconnect from DX database

Parameters: db\_conn - database connection object

options - getopts data.frame

Returns: nothing

### exec\_query

Description: generate a query and execute it. process results afterwards.

Parameters: params - one parameter set from list of parameter sets in input.csv file.

db\_conn - database connection created by connect\_to\_db()

query\_template - query with variables

options - getopts data.frame

test\_period - test period data.frame created by get\_test\_period()

Returns: query results set

### default\_post\_flagged\_processing

Description: default function for processing of flagged query results. it does nothing. overwritten by algorithm if necessary.

Parameters: flagged\_results - query results from flagged query

db\_conn - database connection created by connect\_to\_db()

params - one parameter set from list of parameter sets in input.csv file.

options - getopts data.frame

test\_period - test period data.frame created by get\_test\_period()

Returns: flagged results after processing

### process\_results

Description: process the results of the queries

Parameters: params - one parameter set from list of parameter sets in input.csv file.

db\_conn - database connection created by connect\_to\_db()

flagged\_results - flagged query data results

modulesn\_results - 24-hour healthy query data results

chart\_data\_results - chart data query data results

options - getopts data.frame

test\_period - test period data.frame created by get\_test\_period()

Returns: flagged results after processing

### run\_algorithm

Description: run algorithm for a set of parameters

Parameters: params - one parameter set from list of parameter sets in input.csv file.

db\_conn - database connection created by connect\_to\_db()

flagged\_query\_template - flagged query

modulesn\_query\_template - 24-hour healthy query

chart\_data\_query\_template - chart data query

options - getopts data.frame

test\_period - test period data.frame created by get\_test\_period()

Returns: flagged results after processing

### default\_spark\_load\_data

Description: load parquet files for spark, if needed. default version does nothing. is overwritten by algorithm when needed.

Parameters: db\_conn - database connection created by connect\_to\_db()

params - one parameter set from list of parameter sets in input.csv file.

test\_period - test period data.frame created by get\_test\_period()

Returns: none

### default\_generate\_suppression

Description: default function to generate list of suppressed module SNs. By default it does nothing. it can be overwritten by an algorithm.

Parameters: params - one parameter set from list of parameter sets in input.csv file.

rel\_db\_conn - Reliability database connection

test\_period - test period data.frame created by get\_test\_period()

Returns: list of suppressed SNs

### run\_group\_algorithm

Description: run algorithm for all sets of parameters

Parameters: params - all parameter sets from input.csv file.

options - getopts data.frame

test\_period - test period data.frame created by get\_test\_period()

flagged\_query\_template - flagged query

modulesn\_query\_template - 24-hour healthy query

chart\_data\_query\_template - chart data query

Returns: none

### real\_main

Description: main entry point to algorithms

Parameters: config\_type - type of connection

number\_of\_days - number of days used by algorithm

flagged\_query\_template - flagged query

modulesn\_query\_template - 24-hour healthy query

chart\_data\_query\_template - chart data query

product\_line\_code - default product line code, could be NA

Returns: none

### main

Description: main entry point to R script

Parameters: number\_of\_days - number of days used by algorithm

flagged\_query\_template - flagged query

modulesn\_query\_template - 24-hour healthy query

chart\_data\_query\_template - chart data query

product\_line\_code - default product line code, could be NA

config\_type - type of connection

Returns: none

## main.R

The file main.R defines the actual algorithm. It contains the following.

1. flagged\_query\_template - template query defining the set of flagged instruments.
2. modulesn\_query\_template - template query defining the potential set of healthy instruments.
3. reliability\_query\_template - template query determining the set of instruments which were visited in the last 45 days.
4. post\_flagged\_processing - algorithm-specific function called after the flagged query is execufed.
5. spark\_load\_data - algorithm-specific function for loading parquet files into Spark memory.
6. A call to main() where the parameters are:
7. Number of dayts
8. Flagged query template
9. Modulesn query template
10. Chart data query template
11. Product line code (can be NA).

## RLIB

RLIB contains three file (previously described). Common\_utils.R and main.R which are used in the flat script layout. Adhoc\_common\_utils.R is used by algorithms using the adhoc script layout. Apollo maintains the file common\_utils.R separate from the remaining files in the flat script layout.

## Testing

Since there are two script layouts, hence two testing environments are required. The environments are identical from the Apollo point-of-view. Apollo performs the same operations for the both formats. The files which are contain in the Apollo algorithm zip files changes depending on the script format. See table below:

| **Description** | **Flat Script File Format** | **Adhoc Script File Format** |
| --- | --- | --- |
| Names of files in algorithm Zip file uploaded to Apollo. | main.R algorithm.R | adhoc\_common\_utils.R main.R |
| Names of files created by Apollo for the algorithm to run | config.csv input.csv | config.csv input.csv |
| Utility file maintained by Apollo. | common\_utils.R | common\_utils.R |
| Names of files created by the algorithm | results.csv errors.csv  chart\_data.csv  query.sql | results.csv errors.csv query.sql |
| Names of files which must exist in the directory for the algorithm to run | Input.csv config.csv main.R algorithm.R common\_utils.R | Input.csv config.csv main.R adhoc\_common\_utils.R |
| Command to start R-script by hand. | Rscript main.R | Rscript main.R |
| Command to start R-script when using test dates CSV file | Rscript main.R | Rscript main.R -T |
| Command to start R-script so it only generates query.sql | NA | Rscript main.R -Q |

### Testing with test\_dates.csv File

The test\_dates.csv file contains the following rows. It contains two columns: NAME, VALUE.

|  |  |  |
| --- | --- | --- |
| Name | Value | Description |
| START\_DATE | YYYY-MM-DD | Start date for algorithm |
| END\_DATE | YYYY-MM-DD | Run date (end date + 1 day) |
| MODULESN\_START\_DATE | YYYY-MM-DD | One day before END\_DATE. Use to generate the healthy instrument list. |
| MODULESN\_END\_DATE | YYYY-MM-DD | Same as END\_DATE |
| HEALTHY\_FLAG\_DATE | YYYYMMDD | Same as END\_DATE |

#### Example of test\_dates.csv file for algorithm requiring one day

The algorithms run from midnight to midnight. The range of dates for the algorithm is calculated as the data local time being greater than or equal to START\_DATE midnight (00:00:00) and less than END\_DATE midnight (00:00:00). Lines can be commented out in the any CSV file by using a ‘#’ sign.

NAME,VALUE

#

# START\_DATE,2018-12-01

# END\_DATE,2018-12-02

# MODULESN\_START\_DATE,2018-12-01

# MODULESN\_END\_DATE,2018-12-02

# HEALTHY\_FLAG\_DATE,20181202

#

START\_DATE,2019-06-01

END\_DATE,2019-06-02

MODULESN\_START\_DATE,2019-06-01

MODULESN\_END\_DATE,2019-06-02

HEALTHY\_FLAG\_DATE,20190602

#### Example of test\_dates.csv file for algorithm requiring seven days

NAME,VALUE

# START\_DATE,2018-11-21

# END\_DATE,2018-11-28

# MODULESN\_START\_DATE,2018-11-27

# MODULESN\_END\_DATE,2018-11-28

# HEALTHY\_FLAG\_DATE,20181127

#

START\_DATE,2019-10-11

END\_DATE,2019-10-18

MODULESN\_START\_DATE,2019-10-17

MODULESN\_END\_DATE,2019-10-18

HEALTHY\_FLAG\_DATE,20191018

### Testing So Only Flagged-Query Is Executed

The case may arise where the algorithm is required to only execute the flagged instruments and nothing else required to Apollo. The procedure for running the algorithm in flag-only mode is as follows:

1. Create a directory containing the required files (see table in section for testing).
2. Create an empty file in the directory called “flag\_only”.
3. Start algorithm.

### Query.sql File

All algorithms will generate a file called query.sql which contains all the queries which were executed. This file is very useful since it contains the exact query which was executed via Athena or Spark. Also, if any variables are resolved in the query template, then the query will fail. You can check in the query.sql file for any query containing unresolved variables.

# Algorithms

## Current Algorithms

The current list of supported algorithms is listed in the following table. The table identifies the script format which was used to create the algorithm. Be aware that a few of the algorithms actually use a hybrid version of the flat script format.

| **Algorithm Name** | **Script Format** | **Exceptions to base format** |
| --- | --- | --- |
| alinity\_hc\_waste | flat script format | Overrides pre\_flagged\_processing(), |
| alinity\_waterbath | flat script format | Overrides pre\_flagged\_processing() |
| architect\_cc\_cuvette\_combined | flat script format | Main.R is specific to algorithm.  Overrides post\_flagged\_processing() |
| architect\_dark\_counts\_gt\_avg250 | flat script format | Overrides post\_flagged\_processing() |
| architect\_dark\_counts\_gt\_sd100 | flat script format | Overrides post\_flagged\_processing() |
| architect\_dm\_fail\_cc | flat script format |  |
| architect\_dm\_fail\_ia | flat script format |  |
| architect\_fe\_pressure | flat script format | Overrides post\_flagged\_processing() |
| architect\_wam\_a | flat script format | Main.R is specific to algorithm.  Overrides post\_flagged\_processing(). |
| architect\_wam\_b | flat script format | Main.R is specific to algorithm.  Overrides post\_flagged\_processing(). |
| architect\_wam\_c | flat script format | Main.R is specific to algorithm.  Overrides post\_flagged\_processing(). |
| cc\_cuvette\_integrity | flat script format | Overrides post\_flagged\_processing() |
| cc\_cuvette\_wash | flat script format | Overrides post\_flagged\_processing() |
| ia\_dark\_counts | flat script format | Overrides post\_flagged\_processing() |
| ia\_fe\_pressure | flat script format | Overrides post\_flagged\_processing() |
| ia\_itv | flat script format |  |
| ia\_pipettor\_sample\_syringe\_backlash | flat script format | Overrides post\_flagged\_processing() |
| ia\_process\_jams\_5756 | flat script format | Overrides post\_flagged\_processing() |
| ia\_process\_jams\_5758 | flat script format | Overrides post\_flagged\_processing() |
| ia\_rsm\_pick\_and\_load | flat script format |  |
| ia\_trigger\_pretrigger\_dispense | flat script format |  |
| ia\_vacuum\_leak | flat script format | Overrides post\_flagged\_processing() |
| ia\_vacuum\_pump | flat script format | Overrides post\_flagged\_processing() |
| ia\_vacuum\_sensor | flat script format | Overrides post\_flagged\_processing() |
| ia\_washzone\_aspiration | flat script format | Overrides post\_flagged\_processing() |
| phm\_a3600 | adhoc script format | Uses adhoc common utils. |
| phm\_aps | adhoc script format | Uses adhoc common utils. |

## Mapping input.csv Parameters to Variables in Query Templates

Fundamental to the current R/SQL script design is the ability to substitute into a SQL query the parameter values defined in Apollo for an algorithm or having the parameter value accessible in the algorithm R-script.

A sample input.csv file is shown below.

ALGORITHM\_NAME,PHM\_PATTERNS\_SK,PARAMETER\_NAME,PARAMETER\_VALUE

Alinity IA Optics Dark Count,11111,THRESHOLDS\_COUNT,1

Alinity IA Optics Dark Count,11111,TESTID,10

Alinity IA Optics Dark Count,11111,INTEGRATEDDARKCOUNT\_MAX,543

Alinity IA Optics Dark Count,11111,INTEGRATEDDARKCOUNT\_SD,110

Alinity IA Optics Dark Count,11111,IHN\_LEVEL3\_DESC,Alinity IA Optics Dark Count

Alinity IA Optics Dark Count,11111,THRESHOLD\_DESCRIPTION,Alinity IA Optics Dark Count

In the above file there are two columns PARAMETER\_NAME and PARAMETER\_VALUE. The parameter name serves as the variable name in the SQL query template. See the example below. Notice the parameter names are surrounded by ‘<’ and ‘>’ signs to prevent any accidental clash in names,

select

upper(trim(dxr.moduleserialnumber)) as modulesn,

dxr.productline as pl,

date\_format(max(dxr.datetimestamplocal),'%Y%m%d%H%i%s') as flag\_date,

count(dxr.testid) as num\_testid,

max(dxr.integrateddarkcount) as max\_idc,

stddev(dxr.integrateddarkcount) as sd\_idc

from

dx.dx\_205\_alinity\_i\_result dxr

where

'<START\_DATE>' <= dxr.transaction\_date

and

dxr.transaction\_date < '<END\_DATE>'

and

dxr.integrateddarkcount is not null

and

dxr.integrateddarkcount >= <THRESHOLDS\_COUNT>

and

upper(trim(dxr.moduleserialnumber)) like 'AI%'

group by

upper(trim(dxr.moduleserialnumber)),

dxr.productline

having

count(dxr.testid) >= <TESTID>

and

max(dxr.integrateddarkcount) >= <INTEGRATEDDARKCOUNT\_MAX>

and

stddev(dxr.integrateddarkcount) >= <INTEGRATEDDARKCOUNT\_SD>"

After the substitutions are performs, all the parameters are replaced with the values in the input.csv file.

select

upper(trim(dxr.moduleserialnumber)) as modulesn,

dxr.productline as pl,

date\_format(max(dxr.datetimestamplocal),'%Y%m%d%H%i%s') as flag\_date,

count(dxr.testid) as num\_testid,

max(dxr.integrateddarkcount) as max\_idc,

stddev(dxr.integrateddarkcount) as sd\_idc

from

dx.dx\_205\_alinity\_i\_result dxr

where

'2019-10-20' <= dxr.transaction\_date

and

dxr.transaction\_date < '2019-10-21'

and

dxr.integrateddarkcount is not null

and

dxr.integrateddarkcount >= 1

and

upper(trim(dxr.moduleserialnumber)) like 'AI%'

group by

upper(trim(dxr.moduleserialnumber)),

dxr.productline

having

count(dxr.testid) >= 10

and

max(dxr.integrateddarkcount) >= 543

and

stddev(dxr.integrateddarkcount) >= 110

The start date and end date variables are called ‘START\_DATE>’ and ‘<END\_DATE>’. These values are generated internally and are automatically resolved. The end date is the run date and the start date is the end date minus the number of days required by the algorithm. Also remember, all data are from midnight to midnight.

he healthy list start and end dates are called ‘<MODULESN\_START\_DATE>’ and ‘MODULESN\_END\_DATE>’. These variable generated from the run date minus one day for the start and the run date for the end data

## Charting

### Charting in results.csv

Charting using the results.csv column “CHART\_DATA\_VALUE” is performed in the following ways:

1. Default chart value is 1 for flagged instruments and 0 for healthy instruments, No additional processing is required as long as the MODULESN\_QUERY\_TEMPLATE is defined.
2. Charting is one of the columns from the FLAGGED\_QIERY\_TEMPLATE and a CHART\_DATA\_QUERY\_TEMPLATE is provided. Then the function post\_flagged\_processing() must be overridden to assign the algorithm-specific data column to the results.csv column CHART\_DATA\_VALUE.

### Charting in chart\_data.csv

Chart data in the file chart\_data.csv is currently supported only for WAM A, B and C. The format of this file was previously described. The file main.R must be overridden to calculate the required chart data. There are two columns which are required for the chart data: CHART\_DATA\_VALUE and DATA\_SERIES.

## Adding a New Algorithm

Adding a algorithm requires the following steps when the flat script format is used:

1. Define flagged query template to get list of flagged data. The query must return at the very least: MODULESN, FLAG\_DATE. It can also return product line PL. If so, then the global product\_line\_code variable in the algorithm.R must set to NA.
2. Define modulesn query template to generate a list of healthy instruments (flagged instruments are removed from this list). The query must return the MODULESN at the very least. It cab return a product line as PL.
3. If suppression is enabled, then the global variable use\_suppression must be set to TRUE. The function generate\_suppression() must be overridden in the algorithm.R file. The function returns a data.frame with a column called MODULESN. The data.frame can have no rows if no iinstruments meet the query criteria. Any serial number in this list and in the flagged list will be removed from the flagged list. The list of suppressed instruments is written to the file suppression.csv.
4. Set global parameter number\_of\_days to the number of days required by the algorithm Most algorithms use 1 day or 7 days.
5. Chart data query template is used to generate the charting data in the results.csv file. It is set tp NA if no charting is required, or if the default charting of 1 for flagged and 0 for not-flagged is sufficient. This query must return the columns MODULESN, FLAG\_DATE and CHART\_DATA\_VALUE.
6. Global parameter product\_line\_code is set to the product line code if only one product line is used. It is set to NA if the product line code is returned by the queries.
7. Global parameter config\_type determine how the algorithm connects to the DX database. It is set to “dx” if athena is used and it is set to “spark” for Spark. The adhoc common utils support a third setting of “ida”.

### Customizing Algorithms

There are a few functions which can be customized for each algorithm. These functions allow for special processing specific for each algorithm. The table below summaries the functions.

| **Function Name** | **Description** |
| --- | --- |
| pre\_flagged\_processing | Processing performed immediately age flagged query. |
| post\_flagged\_processing | Processing performed after flagged, healthy and charting queries. |
| Generate\_suppression | Generate list of suppressed serial numbers. |
|  |  |