Contents

[Introduction 2](#_Toc16703764)

[Algorithm Component Files 3](#_Toc16703765)

[Common\_utils.R 4](#_Toc16703766)

[Main,R 6](#_Toc16703767)

[Algorithm.R 7](#_Toc16703768)

# Introduction

The following set of Alinity and Architect algorithms were implemented

using a combination of the statistical programming language R and AWS

Athena SQL.

This is the list of the currently available algorithms:

* architect\_dark\_counts/
* cc\_cuvette\_integrity/
* architect\_dark\_counts\_gt\_avg250/
* cc\_cuvette\_wash/
* architect\_dark\_counts\_gt\_sd100/
* ia\_dark\_counts/
* architect\_fe\_pressure/
* ia\_fe\_pressure/
* architect\_fe\_pressure\_i1/
* ia\_itv/
* architect\_fe\_pressure\_r1/
* ia\_pipettor\_sample\_syringe\_backlash/
* architect\_fe\_pressure\_r2/
* ia\_process\_jams\_5756/
* architect\_fe\_pressure\_samp/
* ia\_process\_jams\_5758/
* architect\_fe\_pressure\_statsamp/
* ia\_vacuum\_leak/
* architect\_wam\_a/
* ia\_vacuum\_pump/
* architect\_wam\_b/
* ia\_vacuum\_sensor/
* architect\_wam\_c/ i
* a\_washzone\_aspiration/

# Algorithm Component Files

Each algorithm is composed of the following list of files:

common\_utils.R

main.R

algorithm.R

The files main.R and algorithm.R are common to all of the algorithms

listed above. The file algorithm.R contains all the code and declarations

which are specific to the algorithm.

## Common\_utils.R

Common utils file contains the following functions which are used in all

the algorithms:

1. reliability\_connect\_to\_db - main entry point for connecting to Reliability database.
2. reliability\_dsn\_connect\_to\_db - entry point to connect to Reliability database using a DSN. Called from reliability\_connect\_to\_db().
3. reliability\_non\_dsn\_connect\_to\_db - entry point to connect to Reliability database NOT using a DSN. Called from reliability\_connect\_to\_db().
4. dx\_connect\_to\_db - main entry point for connecting to DX database.
5. dx\_custom\_connect\_to\_db - entry point to connect to DX database when a Customn connection is used.
6. dx\_saml\_connect\_to\_db - entry point to connect to DX database when a SAML connection is used.
7. spark\_connect\_to\_db - main entry point for connecting to Spark to access DX parquet files and to use Spark SQL via the R package Sparklyr.
8. connect\_to\_db - main entry point to connect to the DX data via Athena, IDA (retired) or Spark.
9. disconnect\_from\_db - disconnect from the DX database for both Athena, IDA and Spark.
10. get\_test\_period - generate sampling period for running the algorithm. The number of days is passed in. Also has the option to get the dates from a separate file for testing.
11. read\_test\_dates - read dates for sampling period from the file test\_dates.csv. This function is used for testing and debugging.
12. read\_csv\_file - read in a CSV file and return as an R data frame.
13. query\_subs = substitute values for variables in query templates.
14. make\_save\_to\_file - pseudo-constructor for save\_to\_file() function. Save data between calls in the captured scope.
15. save\_to\_file - save a data frame to a file.
16. make\_error\_handler - pseudo-constructor for error handling object. errors - error handling object and functions.
17. make\_write\_results - pseudo-constructor for write\_results() function. Save data between calls in the captured scope.
18. write\_results - write flagged and healthy data to a CSV file.
19. empty\_results - return an empty data frame.
20. exec\_query - substitute values for query template variables and execute query.
21. post\_flagged\_processing = default post flagged query processing which does nothing. It is overwritten by each algorithm when necessary.
22. spark\_load\_data - default function to load DX parquet files in to Spark which does nothing. It is overwritten by Spark-based algorithms when necessary.

## 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 are 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 algorithm will overwrite as needed.
      5. Using Reliability database data, suppress any module SN which were recently visited.
      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
15. exit with a status of -1.

## Algorithm.R

The algorithm-specific file algorithm.R 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 - algorithm-specific function called after the flagged query is executed.
8. spark\_load\_data - algorithm-specific function for loading parquet files into Spark memory.