1. Purpose

The purpose of the SomaLogic® DataDelve Tools: Normalize user guide is to provide an installation and operations manual for users.

2. Scope

The scope of this document is to provide the user with a manual to address the installation, dependency needs, data preparation, and operational instructions for the proper use of DataDelve Tools: Normalize.

Contents

3. Overview

DataDelve Tools: Normalize is an analysis pipeline for performing normalization, calibration, and quality control to SomaScan® data.

Key functionality of DataDelve Tools: Normalize command line application includes:

- Enable offsite client sites to perform the SomaScan assay analysis on site without having to send their data to SomaLogic for analysis through emails or online portals.
- Import and check configurations for the system to run an analysis (calibration and normalization) on a sample or plate of samples.
- Run standard or custom analysis to create configurable ADAT files and quality reports.

4. Legal

By downloading, installing, and/or using the DataDelve Tools: Normalize software, you agree to be bound by the following terms.

4.1. License

1. SomaLogic Operating Co., Inc. (“SomaLogic”) grants to the user (“Licensee”) a non-exclusive, non-transferable, non-sublicensable, revocable license (the “License”) to use the DataDelve Tools: Normalize software.
2. “Software” includes the executable computer programs and any related content functionality, or documentation.
3. SomaLogic exclusively retains all ownership title in the Software including copyright to the software itself and any related copyright, trademarks, patents, or other intellectual property rights.
4. The rights and obligations of this License are personal rights granted to the Licensee only and are not transferrable or assignable to any other person or legal entity.
5. Licensee may not modify the Software or cause the Software to be modified, reverse engineered, or decompiled in any manner, in whole or in part.

4.2. Disclaimer of Warranties

- TO THE FULLEST EXTENT PERMITTED UNDER LAW AND SAVE FOR THE RIGHTS GRANTED TO CONSUMERS UNDER APPLICABLE LAWS, THE SOFTWARE IS PROVIDED "AS IS." SOMALOGIC MAKES NO REPRESENTATION OR WARRANTY OF ANY KIND WHATSOEVER TO YOU OR ANY OTHER PERSON RELATING IN ANY WAY TO THE

SOFTWARE, OR ANY OTHER CONTENT OR SERVICE THAT MAY BE ACCESSIBLE DIRECTLY OR INDIRECTLY THROUGH THE SOFTWARE. SOMALOGIC DISCLAIMS ALL IMPLIED WARRANTIES, FOR EXAMPLE THAT THE SOFTWARE, CONTENT, AND SERVICES ARE MERCHANTABLE, OF SATISFACTORY QUALITY, ACCURATE, TIMELY, FIT FOR A PARTICULAR PURPOSE OR NEED, OR NON-INFRINGING. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY SOMALOGIC OR A REPRESENTATIVE SHALL CREATE A WARRANTY.

- SOMALOGIC DOES NOT GUARANTEE THAT THE SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE CONTENT IS ERROR-FREE, RELIABLE, WITHOUT INTERRUPTION OR AVAILABLE AT ALL TIMES. WE DO NOT GUARANTEE THAT ANY SERVICES PROVIDED RELATED TO THE SOFTWARE, INCLUDING ANY SUPPORT SERVICES, WILL BE EFFECTIVE, RELIABLE, AND ACCURATE OR MEET YOUR REQUIREMENTS. WE DO NOT GUARANTEE THAT YOU WILL BE ABLE TO ACCESS OR USE THE SOFTWARE AT ALL TIMES OR LOCATIONS OF YOUR CHOOSING.

4.3. Limitation of Liability

- TO THE FULLEST EXTENT PERMITTED UNDER LAW AND SAVE FOR THE RIGHTS GRANTED TO CONSUMERS UNDER APPLICABLE LAWS, SOMALOGIC WILL HAVE NO OBLIGATION OR LIABILITY (WHETHER ARISING IN CONTRACT, WARRANTY, TORT, INCLUDING NEGLIGENCE, PRODUCT LIABILITY, OR OTHERWISE) FOR ANY DAMAGES OR LIABILITIES, INCLUDING DIRECT, INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL (INCLUDING ANY LOSS OF DATA, REVENUE OR PROFIT OR DAMAGES ARISING FROM PERSONAL INJURY/WRONGFUL DEATH) ARISING WITH RESPECT TO YOUR USE OF THE SOFTWARE, EVEN IF SOMALOGIC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION APPLIES TO DAMAGES ARISING FROM (i) USE OF OR INABILITY TO USE THE SOFTWARE, (ii) COST OF PROCUREMENT OF SUBSTITUTE GOODS AND SERVICES, (iii) UNAUTHORIZED ACCESS TO OR ALTERATION OF YOUR TRANSMISSIONS BY THIRD PARTIES, (iv) THIRD-PARTY CONTENT MADE AVAILABLE TO YOU THROUGH THE SOFTWARE, OR (v) ANY OTHER MATTER RELATING TO THE SOFTWARE.

4.4. Acceptance

- All terms, conditions, and obligations of this License will be deemed to be accepted by the Licensee (“Acceptance”) upon the Licensee’s installation or use of the Software.

4.5. Termination

- This License shall terminate immediately upon expiration of the Term as defined herein, or when Licensee has failed to comply with any of the terms of this License or is in breach of this

License. On termination of this License for any reason, the Licensee will promptly destroy all copies of the Software.

4.6. Term

- The term of this License will begin on Acceptance and will continue for a period of one (1) year, unless an extension is agreed in writing by SomaLogic.

4.7. Governing Law

- This License shall be interpreted according to the laws of the State of Colorado and applicable United States federal intellectual property laws. The Parties agree to submit to the jurisdiction of the courts of the State of Colorado.

4.8. Miscellaneous Provisions

1. This License does not create or imply any relationship in agency or partnership between SomaLogic and the Licensee.
2. If any term, covenant, condition, or provision of this License is held by a court of competent jurisdiction to be invalid, void, or unenforceable, the Parties agree that such provision should be reduced in scope only to the extent deemed necessary to render the provision enforceable, and that the remainder of the provisions in this License will in no way be impaired or invalidated.

If you have any questions, comments, or concerns about this License, please contact us at:

SomaLogic Operating Co., Inc.

Attn: Legal Department

2945 Wilderness Place, Boulder, CO 80301

LegalDept@somalogic.com

5. Installation

- The DataDelve Tools: Normalize application has been built as an executable Python package in GitHub. In order to install this package please obtain a personal access token from SomaLogic. Please ensure that the appropriate executable zip file has been distributed to your local machine. Your executable zip file should have the following name format: <your-os>_datadelvetools.zip. First, decompress the contents of this file into a folder with same name, and then follow the instructions below for your specific system:

5.1. Install On Windows

5.1.1. Install on Windows 10 or 11 (via powershell)

In a Powershell, cd to the unzipped folder containing the datadelvetools executable, called datadelvetools.exe, and from here you can run the tool by providing application commands to the executable datadelvetools.exe. You can add this full path of the executable (for example, D:\<path_to_folder>\windows_datadelvetools\executable) to your Windows PATH variable, open a new Powershell and you can now run the datadelvetools.exe application from any directory. Note if this is done, the directory that the Powershell is running the exe from will save the default DDT_KEY.txt file here (if encryption is enabled and no manual key given). An application command is provided by first specifying the path to the executable and then the desired command, its arguments, and options. This is described in greater detail in the usage section.

5.1.2. Install on Windows 10 or 11 using WSL

The Linux build of the application can be run in Windows using Windows Subsystem for Linux (WSL), which is preinstalled on Windows 10 and 11.

- Open an Administrator Powershell (right-click the Powershell application and select "Run as Administrator")
- Enter wsl --install. This will both enable WSL and install Ubuntu. This may take several minutes, and may require restarting the computer. If you run this command and get WSL help text, it means you already had WSL enabled. In this case, instead run wsl --install -d Ubuntu to install Ubuntu.
- A separate window will open with a terminal and the Ubuntu logo, which says “Installing, this may take a few minutes…” Hit Enter in this window, and create a username and password when prompted. This will be the username and password you use when prompted in WSL.
- In your administrator Powershell, cd to the unzipped folder containing the datadelvetools executable, which should also contain a file called linux_setup.sh. Run this shell script using the command wsl sudo ./linux_setup.sh to begin the process of configuring your system requirements.
- If you are prompted for a password, use the password created in a previous step. If this is your first time using WSL, it will likely install/update a number of packages.
- After the terminal reads "Process Completed", the application is installed and you can close the window.

Once the application has been installed, it is run by providing WSL commands to the executable datadelvetools.exe in an administrator Powershell. *Please note that using WSL, paths use / instead of \, and will begin with /mnt/c/ instead of C:\.* An application command is provided by first specifying the path to the executable and then the desired command, its arguments, and options.

5.2. Install on MacOS

In the folder, you will see a file mac_<your-architecture>_setup.command. Please ensure that the name of the file matches the architecture of your CPU (intel for older Intel-based Macs, or silicon for Macs with Apple Silicon chips like M1 or M2). Right-click on this file, hold the "option" key and click "Open". Click the "Open" button in the pop-up that appears. A terminal instance will open and begin the process of configuring your system requirements. After the terminal reads "Process Completed", close the window.

Once the application has been installed, it is run by providing commands to the executable datadelvetools.exe in the terminal. If this does not work the first time, you may need to right-click and 'option+Open' the executable from Finder once, like with the installer (this should only be necessary once). An application command is provided by first specifying the path to the executable and then the desired command, its arguments, and options.

5.3. Install on Linux

In the folder, you will see a file linux_setup.sh. Run this shell script in a terminal to begin the process of configuring your system requirements. The terminal will read "Process Completed" when the application has been installed.

Once the application has been installed, it is run by providing commands to the executable datadelvetools.exe in the terminal. An application command is provided by first specifying the path to the executable and then the desired command, its arguments, and options. This is described in greater detail in later sections.

5.4. Setup QC Communication

As part of the usage agreement of this tool, SomaLogic communicates some QC data back to an endpoint to monitor that reagents are performing correctly and to help support any issues a user may have with the application. Only SomaLogic control samples (no client data) is collected. For this feature to communicate successfully, a file named upload.txt must be placed in the folder named credentials found with the executable when it was extracted. The file upload.txt must contain the client_id used for authentication and upload in the first line and the client_secret in the second line. These credentials will be used to authenticate your user with SomaLogic and communicate only QC data. If you wish to see the QC data communicated, it can be found in the QC_results in the output directory of any run that is part of the QC tracking. If a key is not provided or the tool is installed on an air gapped machine, these files can be collected and sent to SomaLogic should there be a question about the tool's performance or about experimental results. Please contact your SomaLogic field representative to assist with setting up the QC Communicaton feature.

6. Usage

This command line program will transform SomaScan Agilent Scan Files or raw ADATs into one or more ADAT files with different levels of data standardization applied. With the command line tool successfully installed, the next step is to gather the appropriate workbooks, Agilent files, and references, then fill in the configuration file.

6.1. Analysis Configuration File

All SomaScan studies must have the data processing steps, references, and parameters assigned via a JSON-formatted analysis configuration file. Standard data analysis configurations have pre-defined templates which can be exported by the ddt-normalize application. These templates contain all the necessary reference files and analysis steps to process a set of plates for a specific biological matrix.

Prior to processing a study or run, users should export a report configuration template for the matrix type corresponding to the biological fluid / kit product number for their samples. The `output-json` command should be used generate an appropriate template. See section 7.3 for usage details.

Once a template has been generated, users need to provide study-specific details before running the analysis. Please reference the table in section 7.3 for a list of these inputs. Fields with an "!" before the name require a value to run.

Fields other than those shown in the table in section 7.3 should not be altered from the default values unless directed by SomaLogic support team. Users also should not attempt to deviate from the default analysis steps or values unless advised by SomaLogic.

6.2. Plate vs. Study Workflows

For large studies where all samples cannot fit on a single run, samples must be spread across multiple plates. The ddt-normalize application can be used to run each plate separately and then merge multiple runs together to complete a study.

6.3. Encryption

The ddt-normalize tool caches application data in a sqlite3 database on the local filesystem. By default, this database will be encrypted using 128-bit AES-GCM encryption. To specify the use of a pre-established encryption key or to avoid using encryption, an `--encrypt-key` option is provided for each command. To disable encryption, pass `--encrypt-key 0` to commands that generate

databases (`full, ingest, adat-ingest`). To use a pre-established encryption key instead of a default key, pass the path of the encryption key value as the optional input. Default encryption keys are written to a text file within the user's current working directory under the name DDT_KEY.txt.

6.4. Thread Count

Many commands come with the option `--thread-count`, to check if a command you are using has this option please run `ddt-normalize <command> --help` and read the available options. If no `--thread-count` is provided, the default behavior is to use your computer's number of cpus / 2 as the number of threads to use. You can manually set `--thread-count`, but it must be within 1 and 5 X your cpu count. If a number less than 1 is provided, then the `--thread-count` will be set to 1. If a value over the maxiumum is set, the maximum (5 X your cpu count) will be used instead.

7. Command Line Interface

Once installed, the application can be run from a user's command line terminal (bash, zsh, etc.) uing the command `ddt-normalize <options>`, where the options refer to the varous commands detailed below. These same options can be viewed from the command line by `ddt-normalize -h`, or `ddt-normalize --help`.

7.1. Command Line Options

Command line options:	Description	Usage

output-json	This command produces an analysis configuration JSON file based on the template used for the specific run type. The remainder of the output template JSON configuration file will need to be appropriately filled in in to run the pipeline.
	
	Options for `<template-type>` are `plasma, serum, urine, csf, 2.5%, 10%`, or `20%`
		`ddt-normalize output-json <template-type> <output-directory>`
		Example:
		`ddt-normalize output-json plasma ~/Documents/ddt_templates`
		Help:
		`ddt-normalize output-json --help`

full	Command used to run a full end-to-end analysis. Requires all inputs needed for the analysis. All parameters can be set in the analysis config file. See section 7.3 for configuration file details.	`ddt-normalize full <path-to-the-config-file>` Example:

Command line options:	Description	Usage

	Fields in the config can be overridden by passing command line options. A full description of the possible command line parameters can be found in Table 3.	`ddt-normalize full ~/studies/study_001/input/study_001.json`
		Help
		`ddt-normalize full --help`

dry-run	Tests whether the provided inputs are correct and can be used to run an end to end analysis.	`ddt-normalize dry-run <path-to-the-config-file>`
		Example:
		`ddt-normalize dry-run ~/studies/study_001/input/study_001.json`
		Help
		`ddt-normalize dry-run --help`

adat-ingest	Command used to begin the pipeline using a raw ADAT file. The arguments required are the path to the input ADAT file and the output directory for the resultant database created by the ingestion process. When using the ADAT Ingestion command, the remaining steps in the pipeline - namely `analyze` and `export` - must be run manually.
	An additional optional argument is the path to a
	configuration JSON file. This file can be used to override the analysis_config described in the provided ADAT file's header. A full description of the possible sections of analysis_config are given in this section.
	When using the adat-ingest command there are some limitations on the aspects of the configuration which can be overwritten. The SOMAmer® metadata, Sample metadata, and RFU values are read directly from the raw ADAT file. Therefore, the following fields in the analysis config cannot be overwritten:
	!SOMAmerReferenceSource
	!SampleWorkbooks
	!SampleWorkbooksConfig
	!AgilentDirectory
		`ddt-normalize adat-ingest <path-to-raw-adat-file> <output-directory> [analysis-config]`
		Example:
		`ddt-normalize adat-ingest ~/studies/study_001/output/study_001.adat ~/studies/study_001/output2`
		Help
		`ddt-normalize adat-ingest --help`

Command line options:	Description	Usage

ingest	First part of a two-stage workflow with `analyze` command. This command can be used to import the experimental data and reference materials for a specific analysis into a SQLite DB. All parameters can be set in the analysis config file.
	
	This command can be used to continue running an analysis which failed previously due to incorrect parameter specification. After the run fails, users can correct the parameters provided on the command line or in the configuration file and attempt the analysis again using the partial command.
		`ddt-normalize ingest <path-to-the-config-file>`
		Example:
		`ddt-normalize ingest ~/studies/study_001/input/study_001.json`
		Help
		`ddt-normalize ingest --help`

analyze	Second part of of a two-stage workflow. Executes the analysis pipeline, opecifically performing the different steps for correcting and normalizing RFU values based on the provided configuration. Requires that `ingest` or `adat-ingest` be performed before running this command.
	
	The provided output directory must contain the database file created by an ingest process. The study identifier must also be provided in case there samples from multiple studies for a plate.
		`ddt-normalize analyze <output-dir> <study-id1>`
		Example:
		`ddt-normalize analyze ~/studies/study_001/output study_001`
		Help
		`ddt-normalize analyze --help`

export	Command to export specific ADAT files or SQS report after either successfully performing the ingest or adat-ingest and analyze comands or the full command.
	
	The step name(s) provided to the command must match step names specified in the analysis-config provided to `ingest` or `full`. The only exception to this is the keyword `raw` which can be used to export the initial ADAT which does not appear in any of the step descriptions. Multiple step names can be specified if they are separated with a space (" ") character.
	
	Command used to export QC files. Will fail if successful `ingest` or `adat-`
		`ddt-normalize export <output-directory> <study-id> --adat "<Step name(s)>"`
		Example:
		`ddt-normalize export ~/studies/study_001/output study_01 --adat "anmlSMP"`
		`ddt-normalize export ~/studies/study_001/output study_01 --qc_report`
		Help
		`ddt-normalize analyze --help`

Command line options:	Description	Usage
	`ingest` and `analyze` commands or the full command have not been completed. Note that the dabase file must reside in the output-directory and must be named according to the tools mapping for the study ID. Spaces and dash '-' chararacters resplaced with underscore '_; characters. Study-01 → Study_01.db	

concat-db	This command can be used to combine plates from the same study into a single database if they were run separately using either the `full` command or the piecewise `ingest` or `adat-ingest`, `analyze`, and `export` commands. Combining multiple runs into a single database facilitates the creation of ADAT files with all of the data from the separate runs in a single place. For databases to be successfully merged they must have data from the same study and must use the same configuration (i.e. step configurations and references).
	
	Please note: If the databases are encrypted, the list of source databases must be the same length as the list of encryption keys. If encryption is not used, users must provide "0" as the encryption key for each database to merge. Encrypt keys must be provided as a comma separated array of strings represnting keys paths, or 0. For export to succeed the database must be named based on the study ID. When in doubt use the database name the tool produced for the first plate.
		`ddt-normalize concat-db [--encrypt-keys <...>] <dest-path> <source-path-list1> ... [<source-path-listN>]`
		
		Example:
		
		Where plate_01.db is encrypted by DDT_KEY.txt in the working directory and plate_02.db is not encrypted.
		
		`ddt-normalize concat-db ~/studies/study_001/combined_data/study_001.db ~/plates/plate_01.db ~/plates/plate_02.db –encrypt-keys "DDT_Key.txt","0"`
		
		Help
		
		`ddt-normalize concat-db --help`

7.2. Output

The ***ddt-normalize*** tool will generate ADAT files for each stage of normalization, in addition to an effect size txt file, a sqlite .db file, and a copy of the configuration file.

7.3. Analysis Configuration File Fields

The configuration file is in JSON. Templates can be generated using the export-json command(see Section 7.1). The fields in the following table are user-definable and should be populated for each study.

Field	Value
!StudyId	Study identifier. This value should match the “StudyId” field values in the assay workbook “Assayed Sample List” and “Plate Map” tabs. Client samples (SampleType = Sample) from the run and assay workbook that do not match this value will be excluded from the analysis and study ADAT. This value will be used as a prefix for output files and also appear in the ADAT header in the “Title” field.
!OutputDirectory	Path to the directory where output files will be written
!SampleWorkbooks	List of file paths to each assay workbook in the study. There is one assay workbook per plate.
!AgilentDirectory	List of Agilent scanner file directories.
!AssayRobot	Identifier of the robot on which the plate or study is executed
!EnteredBy	Analyst running the assay
!ExpDate	Date on which the study was run. Must be in the format YYYY-MM-DD.
!ProteinEffectiveDate	This field is meant to provide consistency of field names with SomaLogic produced ADAT files and not used by this software. Please provided a current date in YYYY-MM-DD format.
SiteId	Optional identifier for the site.
Notes	Optional field to describe study-level information.
RunNotes	Optional field to describe plate-level information .
StudyOrganism	Optional study organism identifier
LabLocation	Optional additional laboratory location information

The Analysis Config file ***must*** also include ReportConfig as a top-level field (this cannot be provided as a command-line override). This field contains a nested JSON object that defines the sequence of analysis steps that will be performed, their step-specific parameters, and several other values, all of which will be described below. Users should not attempt to alter the values of these fields without support from the SomaLogic support team.

- ReportConfig
    - analysisSteps: A list of JSON objects, each of which defines a single analysis step and its required parameters. The set of available steps and their parameters will be described below.
    - filter: A place to define filters.
        - proteinEffectiveDate: The date defining when proteins are still effective, in the YYYY-MM-DD format.

- qualityReports: A list of strings naming QC report types to be generated, typically only "SQS Report".
- normGroupReferenceSource: Path to a file defining the normalization group references.

7.3.1. Step-specific config objects

7.3.1.1. Hybridization Normalization

- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "hybNorm".
- referenceSource: Expected/default value is "intraplate", which will tell the code to use values from within the plate as reference. However, this can also be a path to an external reference file, if so desired.
- includeSampleTypes: A list of sample types to include in this normalization. For example, ["QC", "Calibrator", "Buffer"]

7.3.1.2. Intraplate Median Normalization

- stepName: An arbitrary name for this step, used when multiple instances of the same stepType can be added to a pipeline. Suggested value in this instance is "medNormInt".
- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "medNorm".
- includeSampleTypes: A list of sample types to include in this normalization. For example, ["Calibrator", "Buffer"]
- referenceSource: Expected/default value is "intraplate", which will tell the code to use values from within the plate as reference. However, this can also be a path to an external reference file, if so desired.
- referenceFields: A list of fields to use in the reference, usually ["SampleId"].

7.3.1.3. Crossplate Median Normalization

- stepName: An arbitrary name for this step, used when multiple instances of the same stepType can be added to a pipeline. Suggested value in this instance is "medNormSMP".
- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "medNorm".

- includeSampleTypes: A list of sample types to include in this normalization. For example, ["Sample"]
- referenceSource: Expected/default value is "crossplate", which will tell the code to use values from across plate as reference.
- referenceFields: A list of fields to use in the reference, usually ["SampleType"].

7.3.1.4. Plate Scale

- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "plateScale".
- referenceSource: A path to an external reference file.

7.3.1.5. Calibration

- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "calibrate".
- referenceSource: A path to an external reference file.
- tailThreshold: A numerical value defining the tail threshold.

7.3.1.6. Adaptive Normalization (QC)

- stepName: An arbitrary name for this step, used when multiple instances of the same stepType can be added to a pipeline. Suggested value in this instance is "anmlQC".
- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "ANML".
- effectSizeCutoff: A numerical value defining the effect size cutoff to use in ANML calculations.
- minFractionUsed: A numerical value *defining the minimum threshold for fraction of seqIDs used when calculating effect sizes.*
- includeSampleTypes: A list of sample types to include in this normalization. For example, ["QC"]
- referenceSource: A path to an external reference file.

7.3.1.7. QC Check

- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "qcCheck".

- QCReferenceSource: A path to an external reference file.
- tailsCriteriaLower: A numerical value *defining the lower criteria for the tails.*
- tailsCriteriaUpper: A numerical value *defining the upper criteria for the tails.*
- tailThreshold: A numerical value defining the tail threshold.
- prenormalized: A boolean value defining whether or not pre-normalization has happened. Should be "true".

7.3.1.8. Add LOD SOMAmer Metadata

- stepName: An arbitrary name for this step, used when multiple instances of the same stepType can be added to a pipeline. Suggested value in this instance is "AddLOD".
- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "addSomamerMetadata".
- colDataLabel: A column label in the reference file that will be added as SOMAmer metadata. Here, should be "LoDReference".
- referenceSource: A path to an external reference file.
- prenormalized: A boolean value defining whether or not pre-normalization has happened. Should be "true".

7.3.1.9. Adaptive Normalization (Samples)

- stepName: An arbitrary name for this step, used when multiple instances of the same stepType can be added to a pipeline. Suggested value in this instance is "anmlSMP".
- stepType: Defines which type of analysis step the JSON object will create. Here, should always be "ANML".
- effectSizeCutoff: A numerical value defining the effect size cutoff to use in ANML calculations.
- minFractionUsed: A numerical value *defining the minimum threshold for fraction of seqIDs used when calculating effect sizes.*
- includeSampleTypes: A list of sample types to include in this normalization. For example, ["Sample"]
- referenceSource: A path to an external reference file.

7.4. Command Line Mappings to Analysis Configuration Fields

The analysis configuration file uses a standard JSON format with each of the following fields at the top level. If fields are missing from the "analysis-config" file or users wish to override a value in the file, the following arguments (left) can be provided to overwrite or replace the field (right):

Command Line Argument	Report Configuration Field
study-id	!StudyId
output-dir	!OutputDirectory
somamer-reference	!SOMAmerReferenceSource
sample-workbooks	!SampleWorkbooks
agilent-dir	!AgilentDirectory
assay-type	!AssayType
site-id	SiteId
assay-version	!AssayVersion
assay-robot	!AssayRobot
created-by	!CreatedBy
entered-by	!EnteredBy
exp-date	!ExpDate
notes	Notes
protein_effective_date	!ProteinEffectiveDate
run-notes	RunNotes
study-matrix	StudyMatrix
plate-type	PlateType
lab-location	LabLocation
study-organism	StudyOrganism
calibrator-id	CalibratorId

8. Data Transfer

- This application is for local deployment and local processing, there will be no communication of data across any network connections.
- Clients that this application is deployed to are responsible for the safety of their compute environment.
- If PHI is involved clients are expected to use encrypted filesystems.
- The application will only have access to the data provided through the command line interface and configuration files and therefore will only have access to files the active user has access to.
- Users are responsible for placing files produced by this application in secure locations within their storage, if files are placed in shared locations other Users will have access to those files as well.

9. Troubleshooting and Questions

For assistance, contact techsupport@somalogic.com.