Guide to Prepare Data from Network Formation Assay Experiments for Dose-Response Modeling with the ToxCast Pipeline

Amy Carpenter

Table of Contents

## Warning: package 'data.table' was built under R version 4.2.2

## R version 4.2.1 (2022-06-23 ucrt)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 10 x64 (build 22000)  
##   
## Matrix products: default  
##   
## locale:  
## [1] LC\_COLLATE=English\_United States.utf8   
## [2] LC\_CTYPE=English\_United States.utf8   
## [3] LC\_MONETARY=English\_United States.utf8  
## [4] LC\_NUMERIC=C   
## [5] LC\_TIME=English\_United States.utf8   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] stringi\_1.7.8 data.table\_1.14.6  
##   
## loaded via a namespace (and not attached):  
## [1] compiler\_4.2.1 fastmap\_1.1.0 cli\_3.6.0 tools\_4.2.1   
## [5] htmltools\_0.5.4 rstudioapi\_0.14 yaml\_2.3.6 rmarkdown\_2.20   
## [9] knitr\_1.42 xfun\_0.36 digest\_0.6.31 rlang\_1.1.1   
## [13] evaluate\_0.20

# Overview of the process

This document outlines how to transform the recordings from the Microelectrode Array Network Formation Assay into an “mc0” file which can be analyzed with the ToxCast Pipeline R package. This document and the corresponding run\_me\_Template.Rmd was primarily designed to describe the pre-processing steps for the experimental data produced in the EPA Shafer Lab, but it can be generalized for other audiences. Steps or file directories that are specific to the Shafer Lab are noted.

Input files:

* \_spike\_list.csv files created by recording of activity from a microelectrode array with a Maestro
* files that document the meta-data (minimally treatment and dose) in each well (referred to as “MaestroExperiment Log” files)
* (optionally) files that contain the cytotoxicity (LDH and Alamar Blue) experimental and meta data

Intermediate outputs of the pre-processing steps include:

* A csv file containing well-level activity metrics for 17 endpoints across each DIV
* A csv file containing area-under-the-curve values of the well-level activity metrics for 17 endpoints

Background on the scripts/packages:

Initial functions for calculating parameter values from the spike list files were written by Stephen Eglen. This package is available at <https://github.com/sje30/sjemea>. Later, Diana Hall created the package meadq, which builds upon and provides modified versions of some of the functions in sjemea. Mostly recently, the scripts in this repository build upon and provide modifications of the scripts in meadq. Some of the scripts in this repository are edited versions of those in meadq. Packages and scripts are sourced in the run\_me\_Template.Rmd in the appropriate order such that the intended functions from each package are loaded.

# Known issues with the pre-processing workflow

* In the function spkList2list, recordings are truncated 900 seconds *after the first recorded spike* (rather than 900 seconds after the recording began). This choice may be undesirable for recordings with low activity for 2 reasons:
  + If there is a long period of quiescence at the beginning of the recording, activity parameters such as the mean firing rate, burst rate, etc., would be higher than they would be if the entire recording duration was included.
  + The function spkList2list also checks that recordings are not more than 3 minutes short of the target 15 minute recording duration. Thus, if a recording has more than 3 minutes of quiescence at the beginning, the function may flag it as being too short.
* The input source files
  + Background: All experimental and meta data sources files used to pre-process data for the ToxCast Pipeline will ultimately be saved in Clowder (to Amy’s knowledge). With the current pre-processing approach, the file paths to the source files used are recorded in the files\_log.txt. However, these source files could be modified after the pre-processing is completed. Therefore, theoretically the source files should be copied to a stable location before running the pre-processing scripts. However, this seems like it could result in a lot of data copying, especially if the files will need to be copied again to the ToxCast\_data drive. Going forward, work with the ToxCast data team to determine the best path, and check for any modifications to the source files that occurred after pre-processing before sharing with the ToxCast team.
* The ‘srcf’ columns:
  + Background: The mc0 and sc0 files that are loaded into the ToxCast pipeline are expected to include the column ‘srcf’ (read “source file”). This column should contain the name of the file that was used to generate the data
  + However, for the NFA, usually multiple source files are used to generate the numeric and meta data for a single data row
  + Thus, a solution is to package the source files into .zip files (e.g., 1 for each plate for the MEA endpoints). These .zip files would become the new ‘srcf’. This solution was proposed by Jason Brown to address a similar issue for the HCI data.
* Cytotoxicity data extraction - these scripts are configured to extract the blank-corrected values from the Calculations xlsx data files. Generally, template Calculations files are copied from previous groups before being manually updated by the lab technicians for the current group. However, occasionally the lab technicians may forget to update the cytotoxicity data in a Calculations file for a group (in which case the meta data may be correct but the cytotoxicity data values are not). Refer to the section ‘Extract the cytotoxicity data’ for more notes and potential solutions.

# Abbreviations

## Abbrev Defn  
## 1: MCL or Master Chemical List MaestroExperimentLog file  
## 2: NFA Network Formation Assay  
## 3: TCPL ToxCast Pipeline  
## 4: acnm assay component name  
## 5: cndx concentration index  
## 6: rval raw value (present in multi-conc lvl0  
## 7: spid sample ID  
## 8: srcf source file  
## 9: wllq well quality

# Setting up the pre-processing project folder

* Define the project name for the project (usually the name of the compound set followed by the year the experiments were started, e.g. “TSCA2019”). Don’t use any spaces.
* Create a where you want the output to be saved, named by the project name. (For the Shafer-lab, the project folder should be created in this directory: L:/Lab/NHEERL\_MEA/CCTE\_Shafer pre-process for TCPL/MEA\_dev/CCTE\_Shafer\_MEA\_dev\_pre-processing\_projects).

# Prepare notebook section (Shafer-lab)

* Open the pre-processing notebook for the NFA: L:\Lab\NHEERL\_MEA\CCTE\_Shafer pre-process for TCPL\MEA\_dev\Pre-processing\_MEA\_NFA\_for\_TCPL
* Make a copy of the Notes Template section and rename it with the project name
* Use this notebook section to record to-do items, take notes, and save email records as you pre-process the data. Review the suggested usage of this notebook in the “Overview” page.

# Check out the project folder

Familiarize yourself with the experimental project folder. Get a sense of where all of the culture folders are located, how many there are, which cultures contain data for a single- versus multiple-concentration screen, etc.

If this data has already been published, determine which cultures were used for the analysis, and which were excluded (because of quality concerns). See “Experimental Summary of Data” file if present.

# Review lab notebook for well quality notes

* For most projects, there will be a lab notebook with notes from conducting the experiments. Read through this lab notebook for any notes related to the well quality, dosing, data-alignment, etc (usually you can just scan for any hand-written notes)
* Determine if any cultures, plates, or wells should be excluded because of a misdose, contamination, bad culture, etc.
* Also note if any compounds are dissolved in anything other than DMSO
* Enter any well quality notes into the appropriate table, as described below

# Adjust well quality or add well quality notes

**Background**: Every level 0 table processed with the ToxCast pipeline must contain a binary ‘wllq’ column (read “well quality”), where 1 indicates that the data is usable and 0 that the data should not be used. In the ToxCast Pipeline, all data rows with wllq == 0 are excluded from the curve-fitting. In these pre-processing scripts, all data is assumed to have wllq = 1 unless otherwise indicated in the well quality tables (as described below). The well quality tables and pre-processing scripts also include an option to add a well quality note, regardless of whether the wllq is 1 or 0. These well quality notes are not included in the final level 0 table for the ToxCast pipeline; rather, these notes can be used for internal record-keeping and follow-up analyses.

There are 2 ways to set the wllq to 0 or add a well quality note (described below). Either option can be used; choose whichever option is easiest for each wllq situation.

Note that the wllq notes entered in the first table (“[project\_name]\_well\_quality\_table\_by\_well.csv”) will be applied by the function add\_wllq\_by\_well(), which is called by prepare\_parameter\_values() and run\_cytotox\_functions(). This well quality should not be considered final. The wllq notes in the second table (“[project\_name]\_well\_quality\_table\_by\_treatment\_cndx\_culture\_date.csv”) will be applied within the run\_me script AFTER the treatment names and concentrations have been verified. The 2 well quality columns will be merged to produce the final well quality column for the level 0 data for TCPL.

## Update the wllq when the well ID is known

For wells where you know the specific plate, well and (optionally) DIV that are affected: Enter wells that should be excluded from the analysis in the file “[project\_name]\_well\_quality\_table\_by\_well.csv”

For each well quality note, fill out the columns:

## Column  
## 1: affected\_assays  
## 2: date  
## 3: Plate.SN  
## 4: well  
## 5: DIV  
## 6: wllq\_by\_well  
## 7: wllq\_notes\_by\_well  
## 8: wllq\_ref\_by\_well  
## Description  
## 1: Comma-separated list of which assays the well-quality affects, including NFA, AB (Alamar Blue, aka Cell Titer Blue), or LDH  
## 2: Culture date (YYYYMMDD)  
## 3: Plate Serial Number. Should begin with 'MW'  
## 4: Affected well. Should begin with a letter (for the row) and end with a number. If all of the wells on a given plate are affected, enter 'all'.  
## 5: Day in vitro. If all DIV are affected, enter 'all'. (If any DIV other than the standard expected DIV (5, 7, 9, and 12) are affected, create a separate row in the well quality table for each DIV). If the affected assay is a cytotoxicity assay (LDH or AB), this column will be ignored.  
## 6: Well quality (1 = usable, 0 = not usable)  
## 7: Well quality note   
## 8: Well quality reference (i.e., the source from which you learned about the well quality note)  
## Examples  
## 1: For only the Alamar Blue assay: AB; For all assays: NFA,AB,LDH  
## 2: 20201109  
## 3: MW71-7905  
## 4: A4  
## 5: 12  
## 6: 1; 0  
## 7: Contamination; Misdosed; Recording too short; Temperature lower than usual  
## 8: Lab notebook; Readme note

## Update the wllq when the treatment is known

For wells where you know the treatment, concentration index, and culture date that are affected: Enter info in the table “[project\_name]\_well\_quality\_table\_by\_treatment\_cndx\_culture\_date.csv”

\*\*Note that the well quality notes in this table will be added after the data has been transformed to the long format and the treatment and concentrations meta-data has been verified in the run\_me script. Therefore, these wllq notes will not be present in the AUC, cytotox, and parameters\_by\_DIV intermediate output files.

For each well quality note, fill out the columns:

## Column  
## 1: affected\_assays  
## 2: treatment  
## 3: cndx  
## 4: culture\_date  
## 5: wllq\_by\_trt  
## 6: wllq\_notes\_by\_trt  
## 7: wllq\_ref\_by\_trt  
## Description  
## 1: Comma-separated list of which assays the well-quality affects (NFA, AB (Alamar Blue, aka Cell Titer Blue), or LDH)  
## 2: Name of affected treatment (chemical), as it appears in the meta data files  
## 3: Concentration index tested for the given treatment, with 1 = lowest concentration tested and 7 = highest concentration. Note that this should be the concentration index ON A GIVEN PLATE (rather than all concentrations tested for the treatment in the project).  
## 4: Culture date (YYYYMMDD)  
## 5: Well quality (1 = usable, 0 = not usable)  
## 6: Well quality note   
## 7: Well quality reference (i.e., the source from which you learned about the well quality note)  
## Examples  
## 1: For only the Alamar Blue assay: AB; For all assays: NFA,AB,LDH  
## 2: 7126 C7; DMSO  
## 3: 1  
## 4: 20201109  
## 5: 1; 0  
## 6: Precipitate observed, actual concentration is dubious; Cell debris  
## 7: Lab notebook; Readme note

## Notes

* It is okay if there are multiple rows in either well quality notes table affect the same well. (This may occur if e.g., all wells on plate X were affected because the plate was shaken and an individual well on plate X had a contamination). The scripts will collapse these notes and take the minimum wllq for each well.
* Up to this point, if a treatment was observed to precipitate in the MEA plate (i.e., come out of solution), and we did not retest that treatment in another culture that did not precipitate, we have generally left wllq == 1 but included a wllq\_note for the affected treatments/concentrations.

# Determine if any samples IDs need to be registered

**Background**: In order to process the data with the EPA’s ToxCast pipeline, every tested sample of a chemical treatment in a “must be associated with a sample ID (”spid”) registered in ChemTrack. Non-test chemicals (e.g., controls) do not have to have a registered spid if the well type is set to something other than ‘t’ (for test compounds).

Registered sample IDs are usually present in an excel spreadsheet in the project folder (for example, see L:\Lab\NHEERL\_MEA\Project TSCA 2019\EPA\_25092\_EPA-Shafer\_339\_20190722\_key.xlsx). The sample ID column is often named “EPA\_SAMPLE\_ID” or similar.

If the sample IDs for a given project have not been registered (whether the test compounds or control compound samples created in the lab for which we want TCPL to fit a curve), ask the product owner of ChemTrack to register the samples. For each compound, provide the name of the treatment (as it appears in the MaestroExperimentLog), DTXSID (if available), stock concentration, supplier, lot number, and any additional useful information such as the CASN, purity, etc. The ChemTrack product owner is Madison Feshuk ([feshuk.madison@epa.gov](mailto:feshuk.madison@epa.gov)) as of Summer 2023.

# Install required R packages

1. With RStudio, use the command install.packages(“package name”) to install any of the following packages that you do not already have:
2. openxlsx – includes functions for reading .xlsx Excel files
3. data.table – for robust data manipulation
4. ggplot2 – plotting package (not essential, recommended)
5. stringi – string matching package (not essential, recommended)
6. pracma – used in mutual information scripts
7. compiler – used in mutual information scripts
8. gtools – includes useful functions such as ‘asc’ for getting ascii character code
9. Install the package rhdf5 for reading, writing, and opening h5 files. Use the following commands:

if(!requireNamespace("BiocManager", quietly = TRUE))  
 install.packages("BiocManager")  
BiocManager::install("rhdf5")

If it asks to Update all/some/none packages, select all. Additional info: <https://bioconductor.org/install> and <https://stackoverflow.com/questions/15974643/how-to-deal-with-hdf5-files-in-r>

1. Use the command devtools::install\_github("package name") to install the following packages from GitHub:
2. sje30/sjemea
3. dianaransomhall/meadq

Links to the GitHub repositories: <https://github.com/dianaransomhall/meadq>, <https://github.com/sje30/sjemea>

Additional information on installing packages from GitHub: <https://cran.r-project.org/web/packages/githubinstall/vignettes/githubinstall.html>

# Install Git (optional)

If using a Windows machine, download and install Git for Windows (<https://gitforwindows.org/>) in order to interact with the scripts repository via git.

# Running the pre-processing script (run\_me)

## Set up

* Go to a clone of the pre-processing scripts repository (https://github.com/amycarpenter/CCTE\_Shafer\_MEA\_dev\_pre-processing\_scripts) (for the Shafer Lab, a clone can be access a clone of the repository here: L:/Lab/NHEERL\_MEA/CCTE\_Shafer pre-process for TCPL/MEA\_dev/CCTE\_Shafer\_MEA\_dev\_pre-processing\_scripts). Open the Template\_project folder. Copy the the script run\_me\_template.Rmd and the 2 well\_quality\_tables and save them in the project output folder you just created (For Shafer lab, L:/Lab/NHEERL\_MEA/CCTE\_Shafer pre-process for TCPL/MEA\_dev/CCTE\_Shafer\_MEA\_dev\_pre-processing\_projects\<project\_name>). In all 3 files, replace the word ‘template’ with the project name.
* Open the newly saved run\_me script (run\_me\_<project\_name>.Rmd)
* Load the packages in the first chunk.
* Fill out the “Project variables definitions” section. Note that all filepaths must use forward slashes (“/”) rather than backslashes.
  + project\_name - the project name. Should be the same as the folder name created under ‘Setting up the pre-processing project folder’
  + spidmap\_file - (optional) path to file that maps treatment names to registered sample IDs. Preferably, this file should also contain the stock concentrations for each sample
  + spid\_sheet - desired sheet in spidmap\_file (can be the sheet number or the sheet name)
  + project.input.dir - main directory of the experimental data for project, e.g., L:/Lab/NHEERL\_MEA/Project - DNT\_NTP\_2021. This folder is used to direct the scripts of where to search for experimental data files and .txt files that might affect the well quality.
  + scripts.dir - directory containing a clone of the pre-processing scripts the pre-processing scripts (<https://github.com/amycarpenter/CCTE_Shafer_MEA_dev_pre-processing_scripts>). For Shafer lab, default to L:/Lab/NHEERL\_MEA/CCTE\_Shafer pre-process for TCPL/MEA\_dev/CCTE\_Shafer\_MEA\_dev\_pre-processing\_scripts/R
  + root.output.dir - directory where the project\_name folder is located. For Shafer lab, default to L:/Lab/NHEERL\_MEA/CCTE\_Shafer pre-process for TCPL/MEA\_dev/CCTE\_Shafer\_MEA\_dev\_pre-processing\_projects. This defines where the output will be saved.
  + assay\_component\_map\_filename - filename of the csv file containing a map of the assay component names (acsn, aka ‘parameters’) as they appear in the AUC, DIV, and cytotoxicity data files to the corresponding assay component names (acnm) registered in invitroDB. File is present in the pre-processing repository. Defaults to L:/Lab/NHEERL\_MEA/CCTE\_Shafer pre-process for TCPL/MEA\_dev/CCTE\_Shafer\_MEA\_dev\_pre-processing\_scripts/mea\_nfa\_component\_name\_map.csv
  + project.output.dir - directory where output should be saved for the given project. Defaults to file.path(root.output.dir, project\_name)
* Fill out the information to record the version of the pre-processing scripts that will be used for the given project.
  + To verify that there are no uncommitted modifications to the scripts, open Git Bash in the scripts.dir and run git status. Commit any recent modifications and then run git push to sync with the remote.
  + Enter the branch of the pre-processing scripts that will be used for the given project. Should be the ‘master’ branch. (Run git branch to check which branch is currently displayed in the scripts.dir).
  + Obtain the commit date, message, and SHA of the most recent commit to the scripts repository using either of the following methods
    - Navigate to the scripts GitHub repository (<https://github.com/amycarpenter/CCTE_Shafer_MEA_dev_pre-processing_scripts>), click “commits”, then view the most recent commit
    - Open Git Bash in the scripts.dir. Sync with the remote (run git pull origin), then view the most recent commit with git log -1
  + Enter a link to the GitHub repository for the most recent commit. This should be the GitHub link, followed by “/tree/”. For example, <https://github.com/amycarpenter/CCTE_Shafer_MEA_dev_pre-processing_scripts/tree/5c0bf943f9752015ded8e7f54f864b3359e02f10>. As the pre-processing scripts are modified going forward, this link should serve as a stable reference to the version of the scripts used to pre-process the current project. (Ideally, in the future, the NFA pre-processing scripts could be saved as a package. Then, this step could be replaced by simply referencing the package version.)

## Check for READMEs and other well quality notes in project folder

* The script is set up to scan for any files that end in “.txt” within the project.dir.
* First, the script will read the contents of the “README” .txt files. These files are created by the lab technicians and may record well quality notes (which may or may not be also be recorded in the lab notebook). Review the contents of each README file and update the well quality tables as needed (see “Adjust well or add well quality notes” section).
* Second, the script will show any additional .txt files found in the project.dir that are not named “README”. Check any of these files that look like they may contain an important well quality or other note. Any “Overactive electrodes.txt” files can be ignored.

## Identify source experimental and meta data files

Get a list of the group folders from which you want to pull the experimental data. By default, the script will identify all folders in the project.dir that contain a date (formatted as YYYYMMDD). Modify as needed. Confirm you have the expected number of group folders.

Next we want to get the following files from each group folder:

* 1 Calculations or Summary file containing the cytotoxicity data (LDH and AB) (as well as other data)
* 1 MaestroExperimentLog .csv file which contains the meta data (i.e., the treatment and concentration tested in each well) for each plate tested (usually 3 plates tested per culture/group). (Note this this will soon be updated to sheets in the Calculations file, rather than separate .csv files)
* 1 spike\_list.csv file for each of the 4 DIV of recording (DIVs 5, 7, 9, and 12) for each plate

If the group folders are structured as usual, the function ‘get\_NFA\_files\_by\_group’ can be used to get the desired files. However, this function is relatively new and has not been tested on many projects, so it may need to be modified.

Filter the list of all.files as needed. Then, check that the number of files from each group folder matches what you would expect:

* 1 calculations file
* 3 MaestroExperimentLog files (1 for each plate)
* 12 spike list files (4 for each plate)

If the expected number of files are not present in all.files, view the list to determine which files may be missing or unwanted files added. The function ‘basenames()’ may be helpful to display just the names of the files (and not the full file paths). Add additional filters as needed.

Some notes on selecting spike\_list files:

* If there are multiple versions of the spike list files, some of which include data from ‘overactive electroces’ and some of which do not, we generally want the spike list files that includes the overactive electrodes. Check with lab technicians if unsure.
* Note that in the scripts, the recording will be truncated at 900 seconds after the first spike (for standardization). Thus, it is okay if a recording goes over 900 seconds.
* Note if any plates were recorded on any DIV other than 5,7,9,12 (say a recording was made on DIV 10 instead of 9), there is an option within the prepare\_parameter\_values() script that uses linear interpolation to estimate the parameter values on DIV 9 from the recordings on DIV 7 and 10.

Finally, the list of files in all.files will be saved in a file “[project\_name]\_files\_log.txt”. This .txt file will be read by subsequent functions to get the input data files.

## Extract raw data & calculate features

### Convert spike list files to h5 files

This step reads the data in the spike list files, performs some preliminary calculations/summaries, merges in the meta data in the MaestroExperimentLog, and saves all of this information as h5files. One h5file is created for every spike list file. The output will be saved in the project.output.dir.

The primary function is h5\_conversion(). This function calls the function axion.spkList.to.h5(), which is in the script spike\_list\_functions.R. Refer to the roxygen documentation for definitions of each of the arguments of h5\_conversion().

This function will take several minutes to run. As it runs, output will be displayed to the console that summarizes each spike list file. You may see the following message:

You created a large dataset with compression and chunking. The chunk size is equal to the dataset dimensions. If you want to read subsets of the dataset, you should testsmaller chunk sizes to improve read times.

This message indicates that there are more efficient ways to handle the data, but the data itself should not be compromised. So this message can be ignored.

**Notes**: If you have to terminate R part way through this step, the h5file that it was currently working on may be incomplete. Be sure to delete that partial h5file so that it will be recreated when you restart.

**Common issues**:

* If there are no MaestroExperimentLog files that have a culture date and plate.id that match a given spike list file, h5\_conversion will throw an error. Check the spike list file and MaestroExperimentLog (both the file name and the contents) for any mismatches in the culture date or plate.id.

### Calculate 16 parameter values with meadq/sjemea functions

This step calls the function create\_ont\_csv(), which is a wrapper for the function create\_burst\_ont\_Data(). The function create\_burst\_ont\_Data() handles the calculation of 16 parameters (e.g., mean firing rate, burst rate, number of network spikes, etc). Parameters may also be referred to as “components” or “features”. This function takes the h5files as input. The output will be saved in the project.output.dir in a folder called ‘prepared\_data’. One .csv file will be created for every plate. Within a csv file, every data row corresponds to 1 well from 1 recording (DIV).

**Notes**:

* As the function runs, you will see the following message repeated many times:

‘\*\*\* No network spikes found’

This message can be ignored. (Unfortunately, this is produced by a function within either the meadq or sjemea packages and I do not know how to modify it).

* The output csv files will contain ‘ABEfilt’ in the file name. This stands for “Actively Bursting Electrodes filter.” Briefly, this means that, for relevant parameters, the calculations were restricted to the ‘actively bursting electrodes’ (i.e., electrodes with >= 0.5 bursts per minute) or the ‘active electrodes’ (i.e., electrodes with >= 5 spikes per minute) as appropriate for each parameter. This is the desired setting. See feature\_calculation\_notes.md or create\_burst\_ont\_Data() for more details.

### Calculate the normalized mutual information parameter

This step calculates the normalized mutual information. This is an additional parameter developed by Ken Ball. This parameter is calculated separately from the other MEA parameters because it was developed separately (by Ken Ball) from the other 16 parameters (developed by Diana Hall and Stephen Eglen) and because it takes significantly longer to calculate than the other parameters. This function takes the h5files as input. The output will be saved in the project.output.dir in a folder called ‘All\_MI’. One .csv file will be created for every plate. Within a csv file, every data row corresponds to 1 well from 1 recording (DIV).

This is the most computationally-intensive step. Each plate (corresponding to 4 recordings) takes about 10-15 minutes to run.

### Calculate Area-Under-the-Curve values

In order to combine the parameter values from each DIV, we calculate the trapezoidal Area-Under-the-Curve (AUC) value for each parameter value in every well. This step performs that calculation, in addition to some other checks. The function prepare\_parameter\_values() takes the csv files that contain the 16 parameters and the mutual information for each DIV as input. This function 1) merges all of the input files, 2) performs some checks and standardization, 3) merges in the well quality information from the table “[project\_name]\_well\_quality\_table\_by\_well.csv” (using the function add\_wllq\_by\_well.R), and then 4) saves a single .csv with all of the parameter values by DIV. The next function, parameter\_values\_to\_AUC() reads the output file created by prepare\_parameter\_values(), performs some additional checks, then calculates the trapezoidal AUC for each parameter in every well over the DIV.

Refer to the scripts prepare\_parameter\_values.R and parameter\_values\_to\_AUC.R for details on each argument.

**prepare\_parameter\_values() notes**:

* The default expected\_DIVs for this script are c(5, 7, 9, 12)
* NA parameter values are left as NA
* An option is included to add data rows for DIV 2 with a value of 0 for every parameter (see argument add\_DIV2\_values\_of\_0). This option should be set to TRUE to be consistent with all Shafer Lab’s AUC NFA data pipelined in invitroDB to date (July 2023). These data rows will be noted with the phrase ‘not recorded - dummy data at DIV 2’ in the wllq\_notes\_by\_well and the file.name.
  + Background: Early in the development of the NFA, recordings were made on DIV 2, 5, 7, 9, and 12. Eventually, because so little activity was observed on DIV 2, they stopped recording on DIV 2. When the script burst\_parameter\_to\_AUC.R was made (the predecessor to parameter\_values\_to\_AUC.R), the writer decided to add a point at (DIV = 2, value = 0) for every well and parameter value. This choice to impute dummy data at DIV 2 has been the practice in the Shafer lab since before 2019.
* Wllq\_by\_well notes:
  + The well quality in the output tables defaults to 1 unless otherwise specified in “[project\_name]\_well\_quality\_table\_by\_well.csv”. The well quality should not be considered finalized until the well quality notes from the other well quality table are also integrated (“[project\_name]\_well\_quality\_table\_by\_treatment\_cndx\_culture\_date.csv”). (The second well quality table has to be merged later on in the run\_me, after the treatment and concentration is each well has been verified).
  + If some of the rows in the well quality table that include ‘NFA’ in the affected assays do not match any of the data rows in the div\_data, a warning will be given, e.g.

Some rows in [project\_name]\_well\_quality\_table\_by\_well.csv did not match any rows in dat

This warning indicates that there was likely a typo in the meta data in the well quality table or the div\_data (e.g., incorrect culture date or Plate.SN entered somewhere). If this warning is observed and the div\_data\_files includes the entire data set, check the output to the console and check if any of the meta data in the well quality table does not match the intended meta data in the div\_data.

* Rare-but-sometimes-needed option: interpolate values from non-standard to standard DIVs. Usually, recordings are made on DIV 5, 7, 9, and 12. However, occasionally, e.g., due to inclement weather, recordings may be made on non-standard DIV. For example, if there was inclement weather on DIV 9, a set of plates might have been recorded on DIVs 5, 7, 10, and 12. In order to aim to make the AUC value for these plates comparable to AUC values from other plates with recordings from DIV 5, 7, 9, and 12, the parameter values for DIV 9 can be linearly interpolated from DIVs 7 and 10. This is an imperfect solution, but it has been used occasionally in past projects.
* How values are interpolated with this script:
* The function prepare\_parameter\_values() will check for any non-standard DIV. If any non-standard DIV are present and interpolate\_stnd\_DIVs = TRUE, then this function will call the function linear\_interpolate\_DIV() to interpolate the parameter values at the standard DIVs from the DIVs that were recorded. The wllq\_notes and file.name in the resulting table will indicate the data rows that were interpolated. However, note that the linear\_interpolate\_DIV() function was recently updated (July 2023) and has had limited use, so it has not been well-tested and may need to be updated for a given project.
* Note that interpolating values at standard DIVs from non-standard DIVs is not the same as interpolating a value when a DIV recording was missed entirely (i.e., less than 4 DIV recordings were made). See notes on ‘check for missing DIV’ below for how these instances are handled.

**parameter\_values\_to\_AUC() notes**:

* the default expected\_DIVs for this script are c(2, 5, 7, 9, 12)
* In order to calculate the area under the curve, this function sets any NA parameter values to 0. This has been the practice since before 2019; however, the appropriateness of this choice in uncertain for some parameters. Investigation of alternative approaches or alternatives to the AUC may be warranted for some parameters.
* Checks for missing DIVs:
  + Case #1: If the well quality by well is 0 for any DIV for a specific well and parameter, then the well quality by well for the AUC value for that specific well and parameter will also be set to 0.
  + Case #2: In some cases, a DIV recording may have been missed or not saved for a given plate. For example, a given plate may only have data for DIV 5, 9, and 12 (but not DIV 7). In this case, an AUC value *could* be calculated, but the appropriateness of comparing this AUC value to AUC values from plates with recordings on all of the expected\_DIVs is uncertain. In some past projects (e.g., ~2014) with some missing DIV recordings for a single plate, parameter values from the other 2 plates tested in the same culture were used to generate filler data. Specifically, the medians of wells that tested the same treatment and dose were used to generate filler data for each well of the plate with the missing DIV. The script estimate\_missing\_DIV.R was used to do this. However, this script has not been maintained and this practice is not optimal. Therefore, in the current implementation of parameter\_values\_to\_AUC(), the well quality by well will be set to 0 for the AUC value for any wells/parameters that do not have a value for every DIV in expected\_DIVs.

### Extract the cytotoxicity data

The function run\_cytotox\_functions() will extract the blank-corrected fluorescence and optical density values for the LDH and Alamar Blue assays for the NFA. The functions createCytoTable2() and createCytoData() are called to do the leg work of extracting the data. Input data files are ‘Calculations’ .xlsx files. Data is read from the sheets “LDH” and “AB” (or common synonyms). Refer to roxygen documentation for more information on each function.

You may see the following message:

The following rval’s are NA, but wllq\_by\_well == 1. Wllq\_by\_well will be set to 0

In this case, if the NA rvals are unexpected, check the source file (and possibly lab notebook) to determine the cause. Possible causes include:

* Misalignment in the Calculations files (in which case the structure of the xlsx needs to be fixed)
* Plate was not ran for given assay (in which case, you can update the well quality table with any explanatory wllq\_notes and re-run run\_cytotox\_functions())

**Notes**:

* Any negative blank-corrected values are set to 0 in the function createCytoData()
* For any NA blank-corrected values, the wllq\_by\_well is set to 0 and the value is left as NA.
* The cytotoxicity data extraction functions are hard-coded to only work for standard 48-well (6x8 plates). Functions will need to be modified if plates with different dimensions are used.
* The function createCytoData() works by anchoring the read frames for the chemical, concentration, and blank-corrected values based on certain hard-coded tag phrases. Therefore, if the spelling of any of these key phrases changes, the the function will need to be modified. See the function documentation for more details.
* The function createCytoTable2() will read data from a maximum of 3 plates. This restriction was made to prevent tripping up the function in case the tag phrase “Chemical” appears in additional data rows that do not indicate the start of a data chunk for a given plate.

**Known issues - potential for undetected incorrect data if raw files are not hand-copied by lab technicians**:

The values in the Calculations .xlsx files are updated by hand from the raw readouts of the AB and LDH data. Usually, for each group, the Calculations file will be copied from a previous culture or a template folder to serve as a starting point. **However, sometimes the lab technician may forget to manually copy-paste the data from the raw xls files to the Calculations data file for a group.** To ensure that this is not the case, you could manually check every raw data file to make sure that the data values have been pasted into the Calculations file. Example machine-produced raw data files:

* LDH (contains data for 3 plates per file): L:\_MEA- DNT Test Set 2021 NTP\20210929\_NFA\_DNT 2021\_Group 3Cytotoxicity\20210929\_20211011\_DNT 2021\_G3 G4\_LDH.xls
* Alamar Blue (contains data for 1 plate per file): L:\_MEA- DNT Test Set 2021 NTP\20210929\_NFA\_DNT 2021\_Group 3Cytotoxicity\20210929\_20211011\_DNT 2021\_G3\_75-9306.xls

However, manually checking every file would be very cumbersome. At the very least, check with the lab technicians before extracting the data if, to their memory, they have copied the AB & LDH data to the Calculations file for every plate and group. A better long-term solution may be to write functions that read the raw data from the machine-produced raw data files directly (rather than the Calculations files). Then the raw values could be corrected to the average of the blank wells within the R scripts (usually G1, G2, and G3, or whichever wells are labelled as “Blank”). However, there are 2 possible downsides to this approach:

* Sometimes, based on the expert-judgment of the lab technicians, not all of the blank wells are used to perform the blank-correction or blank wells from another plate are used if the blank wells from a given plate look off. Therefore, if you create functions to read the raw data directly, you should also either communicate with the lab technicians to identify plates where different blank-control wells are used to correct the raw data values or develop standardized cutoffs for blank well acceptability.
* There may have been a dosing error (e.g., doses in 2 columns switched) that only affects one of the cytotoxicity assays, and not the MEA data. For those cases, we may need to maintain separate meta-data documentation for the LDH, AB, and MEA plates.

## Prepare the level 0 table

The function tcpl\_MEA\_dev\_AUC() will read in the parameter values saved in the AUC, DIV, and cytotoxicity data files, merge and transform the data into 1 long-format data table, and do some standardization/cleaning to the table. This function will return a single data table that can be further cleaned in the run\_me script before it saved as a .RData file.

Note that the output data files will contain both the parameter values by DIV (as indicated by the suffix ‘DIV’ in the acnm (assay component source name)) as well as the AUC values (as indicated by no suffix in the acnm). Dummy-data added at DIV 2 with the function prepare\_parameter\_values() will be removed (i.e., any data rows that contain the phrase ‘not recorded - dummy data at DIV’ in the wllq\_notes\_by\_well or the file.name will be excluded).

The function tcpl\_MEA\_dev\_AUC() will check that the MEA (AUC and DIV) and cytotoxicity input data files have the same columns. If not, it will give a warning message. Address any important columns that need to not be NA in the run\_me script (e.g., the units for the cytotoxicity data usually need to be defined later on in the run\_me script).

The output of tcpl\_MEA\_dev\_AUC() has almost all of the columns that are needed for the ToxCast pipeline level 0. The purpose of the subsequent steps is primarily to check the data for any typo’s, misalignments, missed or added data, etc, as well as to create the remaining columns for the level 0 table.

Most of the checks performed are documented in the run\_me script. A few additional notes on some steps are provided here. Throughout the process, refer to the lab notebook and talk to the lab technicians whenever their seems to be an issue with the meta-data.

## Recommended checks and cleaning

### Check treatment labels

The main purpose of these checks is to try to catch any typo-s in the treatment meta data.

Notes on vehicle control well labels:

Usually, every well in column 2 of the MEA plate contains vehicle control wells. These are usually identifiable by the concentration being equal to 0. In the MaestroExperimentLog, the actual vehicle used will be identified in the treatment column (usually DMSO, sometimes Water or EtOH (ethanol)). In some cases, the vehicle control may not be indicated (even though the concentration in the well is 0). For the cytotoxicity data, the vehicle control may not be indicated (i.e., the treatments in column 2 may have dose == 0 but treatment labelled as the test chemical). The treatment label of vehicle controls used in these cases can be extrapolated from the vehicle controls in the same wells of the MEA data. If unsure, check the lab notebook or talk to the lab technicians.

### Check concentrations and units

The main purpose of these checks is to try to catch any typo-s in the concentration (dose) meta data.

Note that sometimes the concentrations may differ between the cytotoxicity and MEA data files for a given well. If these discrepancies in concentration are notable (I usually look for differences > 0.005 log10-uM), we want to determine which data files contain the correct concentrations. Steps to identify and resolve these instances are suggested in the run\_me.

### Note the vehicle controls per plate

In the ToxCast Pipeline, the same-plate vehicle control wells are used to normalize the data (based on the median of well with wllt == ‘n’ on a given apid). Usually, these are all DMSO wells, but occasionally we use Water or Ethanol.

Check here if there are multiple vehicle control types on a given plate. In that case, we may want to split the apid to separate the wells by vehicle control for the normalization step.

Historically, based on a handful of occurrences, we usually have not found a notable difference between the activity in the DMSO, Water, and Ethanol vehicle control wells. Therefore, we have not segregated wells on a plate based on the vehicle control in the normalization step up to this point (July 2023). But, as more data on different vehicle controls is collected, we may want to reconsider the normalization for plates with multiple vehicle controls.

### Check for NA rvals

None of the raw values for the AUC, LDH, or AB assay components should be NA at this step (unless wllq == 0). If there are any NA rval’s, check back through the input files/intermediate output files to see where the NAs were introduced.

For the DIV-specific assay components, some NAs will likely be present. Up to July 2023, the NAs have been left in the level 0 table. Any NA data points will be ignored from the curve-fitting in the ToxCast Pipeline. In the future, we may want to consider imputing a value for the NAs (e.g., 0) for some of the DIV-specific assay components.

### Data summaries (verify results with others)

The purpose of this section is to generate some high level summaries to share with others so that they can confirm the data contains what is expected to be present.

Some things that could be checked visually in the summary plots:

* **Activity vs plate/culture\_date**: Does the activity for some plates/cultures appear drastically different than others?
* **Activity vs concentration**: Does the frequency of data points near 0 generally increase as the concentration increases?

## Save & verify the results with others before sending to TCPL

Knit the run\_me rmarkdown to document the pre-processing. The final output data file ([project\_name]\_for\_tcpl\_lvl0.RData) is ready for processing with TCPL (other than “Final notes”, below). The data summaries can be shared with the lab to confirm the final result.

Some things that could be confirmed with the lab technicians:

* expected treatments are present with wllq == 1 (none missing, none added)
* expected repeated treatments are present with wllq == 1 (none missing, none added)
* expected concentration ranges for each treatment

Optimally, have people familiar with the biology look at the data as well. Create additional visualizations and/or summaries and/or csv files as needed to share the output.

# Final notes

(Shafer lab) Update the status of the project in the file L:/Lab/NHEERL\_MEA/CCTE\_Shafer pre-process for TCPL/MEA\_dev/CCTE\_Shafer\_MEA\_dev\_pre-processing\_scripts/MEA\_dev\_status\_by\_project.xlsx. (This file may be useful for keeping track of whether projects have been pre-processed and/or pipelined into invitrodb).

Final things to do before running writing the dat to level 0 for TCPL:

* Remove the data rows associated with acnm that are specific to individual DIV (i.e., where the acnm contains “\_DIV”) (unless you want to pipeline those assay components)
* Make sure that none of the data for the new lvl0 table has been pipelined into invitrodb before and that none of the input data is duplicated. (Some experimental groups may have tested chemicals from multiple projects, in which case the experimental data files may be duplicated in different project folders on the L drive. In these cases, we want to make sure that we only pipeline the data once. Merge the new lvl0 data with the existing lvl0 and check that there is only 1 data row per apid-rowi-coli-acnm combination).