CeMM

SCIENCE IS OUR MEDICINE

Isogenic CPA pipeline overview: Feature extraction & Data pre-processing

R.H. Amanda Ng

Last updated: 11th December 2023

Note: The isogenic CPA pipeline here has been designed for use in the Winter lab, CeMM. The instructions here should assist in the implementation of the isogenic CPA pipeline elsewhere as well. ☺

Research Center for Molecular Medicine
of the Austrian Academy of Science

www.cemm.at

Using this set of slides

This set of slides provides descriptions on how to carry out the feature extraction using *CellProfiler* and *cellpose*, and data pre-processing to obtain morphological profiles from the isogenic CPA. Parts of it can also be applied to the CPA or other morphological profiling approaches.

The paper describing the isogenic CPA can be found here: https://doi.org/10.1021/acschembio.3c00598. I recommend reading this paper to get an overview and the intuition behind the workflow for the isogenic CPA.

This set of slides is best used alongside the implementation of scripts/notebooks that can be downloaded from the github repository associated to the isogenic CPA here: https://github.com/GWinterLab/IsogenicCPA.

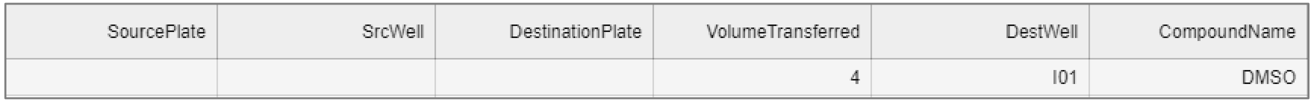
I have tried to annotate the scripts/notebooks extensively with descriptions in this set of slides as well that would hopefully help you/the user implement the isogenic CPA. ☺

1 | Feature extraction

CeMM

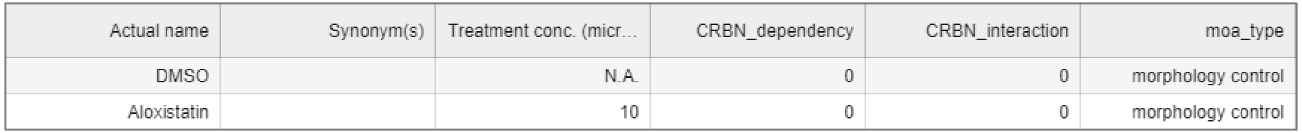
Prepare the following CSV files prior to feature extraction (with examples)

- transferlist.csv # From PLACEBO (screening facility)



SourcePlate	SrcWell	DestinationPlate	VolumeTransferred	DestWell	CompoundName
			4	I01	DMSO

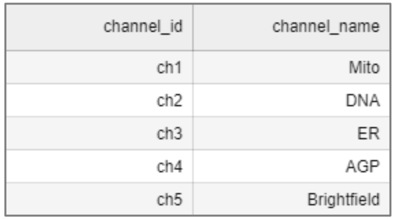
- drug_metadata_sheet.csv # Based on the treatments you are using and downstream analysis goals



Actual name	Synonym(s)	Treatment conc. (micr...	CRBN_dependency	CRBN_interaction	moa_type
DMSO		N.A.	0	0	morphology control
Aloxistatin		10	0	0	morphology control

CRBN_dependency: Known CRBN dependency of the drug. 0 = NOT dependent, 1 = dependent, NaN = unknown
CRBN_interaction: Known or measured CRBN interaction using the fluorescence-polarisation assay.
    0 = NO interaction, 1 = interaction
moa_type: Descriptor for mechanism of action for characterised drugs. Can be: morphology control, protac, inactive protac, mg, inactive mg, inhibitor

- channel_annotation.csv # Based on your imaging set-up

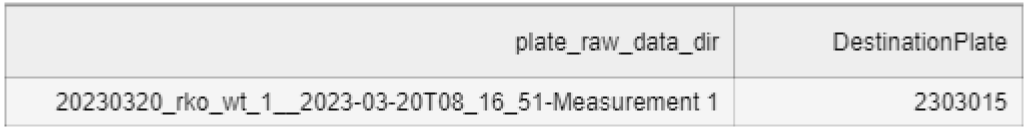


channel_id	channel_name
ch1	Mito
ch2	DNA
ch3	ER
ch4	AGP
ch5	Brightfield

← channel_id here refers to the "ch{number}" in the filename of the raw images (.tiff files)
← channel_name refers to the nickname of the channel that will be used by the CellProfiler pipelines

Prepare the following CSV files prior to feature extraction (with examples)

- `plate_annotation.csv # Based on the file names you have used and what they map to on the barcode`
  This file is important for downstream analysis. The destination plate ID allows you to figure out what images are associated to a specific barcode. Additionally, python has a path length limit by default that can screw up the access of files by python-based scripts. Other languages may have similar defaults. In other words, the path to a file cannot be too long. Plate IDs are typically short and unique, so I designed the pipeline to name the relevant files by the destination plate ID (e.g. 2303015) instead of a user-specified ID (e.g. 20230320_rko_wt_1).



plate_raw_data_dir	DestinationPlate
20230320_rko_wt_1__2023-03-20T08_16_51-Measurement 1	2303015

CeMM

Instructions for running the feature extraction

1. Create a parent directory for storing all the data pertaining to the Cell Painting assay run and a sub-directory for each cell line.

```
Parent_dir_for_all_data
|
+-- cell_line_dir_01
    |
    +-- raw_data_dir
```

2. Export the raw data from the Opera Phenix to the "`raw_data_dir`" for the appropriate "`cell_line_dir`".

```
    :
    +-- raw_data_dir
        |
        +-- raw_data_from_opera_phenix
```

3. Add the following CSV files to the parent directory.

```
Parent_dir_for_all_data
|
+-- drug_metadata_sheet.csv # Details on the treatments in use
+-- transferlist.csv # Details on what drug was dosed into the well and plate (provided by PLACEBO)
+-- channel_annotation.csv # Details to map the channel number to the appropriate channel
+-- plate_annotation.csv # Details on the names you used for the plate and what the plate barcodes are
+-- :
```

4. Run the feature extraction pipeline (parts A, B and C).

CeMM

Command to execute: Part A

```
# Part A
parent_dir="full path to the parent directory for the CPA run"
cell_line_dir_array=("cell line directory 1" "cell line directory 2" "cell line directory 3")
a_path=/research/lab_winter/users/ang/cell_painting_assay_pipeline/01_feature_extraction/DEV_a_cpa_featu
re_extraction.sh
mode=non-test
transferlist="full path to the transferlist.csv"
channel_annotation="full path to the channel_annotation.csv"
Plate_annotation="full path to the plate annotation.csv"
for dir in "${cell_line_dir_array[@]}"
do
  printf "${dir}\n"
  raw_data_dir=${parent_dir}/${dir}/raw_data_dir
  output_dir=${parent_dir}/${dir}/output_dir
  sbatch ${a_path} ${mode} ${raw_data_dir} ${transferlist} ${channel_annotation} ${plate_annotation}
${output_dir}
done
```

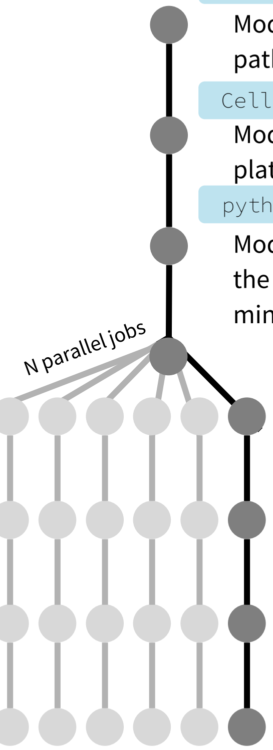
Command to execute: Part B

```
# Part B
parent_dir="full path to the parent directory for the CPA run"
cell_line_dir_array=("cell line directory 1" "cell line directory 2" "cell line directory 3")
b_path=/research/lab_winter/users/ang/cell_painting_assay_pipeline/01_feature_extraction/b_cpa_feature_e
xtraction.sh
for dir in "${cell_line_dir_array[@]}"
do
  printf "${dir}\n"
  output_dir=${parent_dir}/${dir}/output_dir
  sbatch ${b_path} ${output_dir}
done
```

Command to execute: Part C

```
# Part C
parent_dir="full path to the parent directory for the CPA run"
cell_line_dir_array=("cell line directory 1" "cell line directory 2" "cell line directory 3")
c_path=/research/lab_winter/users/ang/cell_painting_assay_pipeline/01_feature_extraction/c_cpa_feature_e
xtraction.sh
for dir in "${cell_line_dir_array[@]}"
do
  printf "${dir}\n"
  output_dir=${parent_dir}/${dir}/output_dir
  sbatch ${c_path} ${output_dir}
done
```

Details on how the feature extraction pipeline works



`python`

Module 1: Generate the metadata file containing the image filename and path details, treatment, etc. for CellProfiler pipelines.

`CellProfiler pipeline`

Module 2: Calculate the illumination correction matrix for each channel per plate.

`python`

Module 3: Update the metadata file with the filename and path details of the illumination correction matrices and split the metadata file into *N* mini-metadata files.

`CellProfiler pipeline`

Module 4b: Prepare the DNA and AGP channel images for nuclei and whole cell segmentation by cellpose.

`cellpose`

Module 5b: Nuclei and whole cell segmentation.

`python`

Module 6b: Update the mini-metadata file with the filename and path details of the segmentation masks.

`CellProfiler pipeline`

Module 7b: Feature extraction from the images described in the mini-metadata file

The feature extraction pipeline is split up into 7 modules that can be found in 🗀 "dependency_scripts".

Part A runs modules 1 to 3. Part B submits N jobs for modules 4b and 5b. Finally, Part C submits N jobs for modules 6b and 7b. The scripts for these parts can be found in 🗀 "01_feature_extraction".

N = 20* by default and is defined by `number_of_batches` **in part A.**

All outputs will be in the cell line specific directory under 🗀 "output_dir"

CeMM

*I changed the default from 10 to 20 on 21 March 2023 to manage more images.

2| Data pre-processing

CeMM

Instructions for data pre-processing

1. Create a sub-directory within the parent directory for all data pertaining to the CPA run. This sub-directory will be where the outputs from data pre-processing will be output to.

2. Within the parent directory, create a notebook for checking image quality, profile assembly and feature selection (see 🗀 "02_feature_analysis" > "Section_1_and_2.html" for an example). The image quality check requires the most supervision and adaptation to the measurements in "Image.csv", so there is no template pipeline.
   Profile assembly does not require tweaking and can be done by using the `profileAssembly(…)` function from the example notebook (just copy and paste).
   Feature selection may require tweaking though. Feature selection is done through the `SelectFeatures(…)` function. Tweaking can be done on the "global" and/or "treatment-ccentric" feature selection strategies. E.g. I could tweak the treatment-ccentric feature selection to allow for more features to be selected with a lower `vote_threshold` to allow for Thalidomide to have at least 1 feature selected.

The general functions required are in `post_feature_extraction_modules.py`.

Details on: Semi-supervised image quality checks

Step 1: The output CSV files from module 7b are in sub-directories numbered from 1 to *N* (based on the number of batches). A unique image identifier is added to all CSV files across batches as "UpdatedImageNumber". Only the Image.csv file across batches is merged.*

Step 2: To check for seeding or staining issues across the plate, the cell count and median channel intensity is plotted as a heatmap formatted as a 384-well plate.
+ retrieve sample images from problematic wells

Step 3: The cell count and median channel intensity is plotted on a per site basis as a KDE plot to identify any outliers.
+ retrieve sample images from problematic sites

Step 4: Flag low quality images with Isolation Forest. The "bad" images are visualised on a 2D Image Quality PCA and as a stacked barplot. OR use some other method for flagging low quality images (e.g. using *PercentMaximal* or *PowerLogLogSlope*).^
+ retrieve sample of bad images^

Step 5: Flag treatments that induce supernumerary nuclei, as the current downstream analysis pipeline assumes that each cell only has one nuclei during profile assembly.^

Step 6: As the nuclei and whole cell segmentation is done independently, there may be discrepancy in nuclei and cell counts. This discrepancy is calculated and plotted in a KDE plot per site. Images with high discrepancy are sampled for retrieval and the segmentation masks are overlaid.^

Proceed to profile assembly and feature selection

*Step 1 can be implemented in the jupyter notebook (slower) or as a non-interactive job on the CeMM cluster (faster) using
`modify_featureExtractionCSV_byCell.sh`
^Steps 4 to 6 are a rough guideline. You can shuffle the steps around or add additional steps where necessary depending on the artefacts observed.

Details on:
Profile assembly and feature selection

After semi-supervised image quality checks

`profileAssembly(…)` For each batch per cell line, the “Cells.csv”, “Nuclei.csv” and “Cytoplasm.csv” is merged according to “UpdatedImageNumber” and the corresponding parent-child relationships. The features per cell is then aggregated on a per site basis by taking the median of all cells. Finally, the data is merged across batches and cell lines (with the “Metadata_Cell” information) as the morphological profile.

`SelectFeatures(…)` Carry out global and/or treatment-ccentric feature selection. The parameters for these feature selection approaches can be adjusted. For treatment-ccentric feature selection, parameters should be adjusted where possible such that thalidomide is retained as a treatment.

End of data pre-processing
(Proceed to interpreting the morphological profiles using the features selected.)

Further details on: Feature selection (1/3)

Feature selection is done with the `SelectFeatures(…)` function, which can be quite daunting to unpack just by looking at the code in 📜 `post_feature_extraction_modules.py` (which you should also do to get a full grasp of it). I will try to summarise what the function does here.

There are three main feature selection methods that can be used in combination for different feature selection strategies:

- `_select_baseline(…)`
  This function discards irrelevant functions that are constant across all treatments, discards functions which have a median absolute deviation (MAD) of 0 in DMSO controls (which will complicate the calculation of robust Z score), convert the feature mesaurements to robust Z score by cell line context and finally discard features with "NaN" or "Inf".
- `_select_decorrelateByMAD(…)`
  This function discards features with are redundant by (1) calculating the variability of each feature, (2) ordering the features by variability and (3) selecting the first feature and iteratively compare the subsequent features with the selected feature(s) and only selects features with a pearson's correlation < 0.8 between it and prior selected features. This function uses robust Z score which is dependent on cell line context, so this function is ONLY implemented on a per cell line basis.
- Selection of features which correlate with the protein (CRBN) levels in cell lines. (This selection is done within `_select_byTreatment`(…), but there is no function that only executed this selection.)

CeMM

Further details on: Feature selection (2/3)

The three main feature selection methods can be used for different feature selection strategies. Broadly speaking, there are two feature selection strategies I have implemented in `SelectFeatures(…)`:

1. “Global”
   “Global” feature selection refers to a treatment-agnostic form of feature selection and it can be implemented in two ways (both ways can also be used):
   - Only one one cell line context
     Only the morphological profile of one cell line is used in `_select_deorrelateByMAD(…)` and the features selected are used. This approach could be used if you are only interested in one cell line context and you wish to compare the morphological data between treatments.
   - On all cell line contexts
     `_select_deorrelateByMAD(…)` is implemented on each cell line separately and features consistently selected by cell line contexts are used via the `_select_consensus(…)` function. I have yet to figure out what an appropriate use case would be; I just wrote this function as an option for future use.
2. “Treatment-centric”
   As implied by its name, this strategy involved selecting feature on a per treatment basis. The morphological profile of all cell lines is handled one treatment at a time. The correlation between the feature measurement and the cell line status (e.g. CRBN KO, CRBN WT and CRBN OE) is calculated as Kendall rank correlation coefficient, tau, and the significance of the correlation is calculated as a p-value. Significant features are selected. These significant features are then weeded out using _select_consensus(…). These steps are repeated for all treatments by `_select_byTreatment(…)`.

Further details on: Feature selection (3/3)

The `SelectFeatures(…)` function is formatted for use in a , perhaps, unusual way:

```
SelectFeatures(
            concatProfile_path,
            prefix = “”,
            corr_threshold = 0.8,
            verbose = True,
            output_dir = “Default”
            param_global = ({                                  # ← The param_global dictionary allows the user to
                        “skip”: False,                         change the global feature selection approach and
                        “get_consensus”: “Both”,               their parameters. You may even choose to skip this
                        “vote_threshold”: 0.5                  feature selection step.
                        “global_cell”: “rko_wt”
            }),
            param_treatment = ({                               # ← The param_treatment dictionary allows the user
                        “skip”: False,                         to change the parameters of the treatment-centric
                        “keep_byProduct”: True,                feature selection You may even choose to skip this
                        “description”: “”,                     feature selection step.
                        “ordered_cells”: [“rko_ko”, “rko_wt”, “rko_oe”],
                        “alpha”: 0.05,
                        “vote_threshold”: 0.5 }) )
```