**Guidelines for bbCARMEN Analysis**

1. **Image Acquisition via fluorescence microscopy**
   1. Capture a sequence of five images in the following order:
      1. Red
      2. Green
      3. Yellow
      4. Blue
      5. UV
   2. Consolidate all five channels into a single composite image. Adherence to the given naming scheme will facilitate seamless integration with the analysis script. However, this is not obligatory; adjustments to Metadata and NamesAndTypes within the pipeline can compensate for deviations.
2. **Image Processing with CellProfiler**
   1. Initialize CellProfiler and import the pre-existing pipeline.
   2. Remove any pre-loaded images and upload new datasets from your storage device.
   3. Extract pertinent metadata via the dedicated metadata tab.
   4. Specify all output directories:
      1. For modules related to "Save Images," designate a new folder within your local storage to optimize space utilization.
      2. Store all CSV outputs in the designated experimental output directory on your institutional computer.
   5. Initiate the image analysis.
      1. The processing duration may vary between 20 to 30 minutes, depending on the size of the experiment.
3. **Preparing the Output Directory for Jupyter Notebook Execution**
   1. Include the following files in the directory:
      1. **Carmenscripts.py** — A Python script originally developed by David and subsequently converted into a .py file.
      2. A Jupyter Notebook (**.ipynb**) file to execute the aforementioned script.
      3. **Classes.csv** — A CSV file mapping color codes to specific viral names. This file is presently static but may require updates when new viral strains are introduced. In the script, each of our colors use a 3 letter code to designate the relative levels of the colors Red, Yellow, and Green – so you may see something like “MNL” which means medium Red, no Yellow, and low Green.
      4. **Samples.csv** — A CSV file assigning descriptive names to each image. This file necessitates updates for each new experiment so that the proper names of the condition are shown on the heatmaps.
      5. v. **ternary\_transform.csv** — Generated upon utilizing the widget to specify cluster assignments; records any deviations from default parameters.
4. **Execution of the Jupyter Notebook**
   1. Initialize the script by executing the first cell.
   2. Optional parameter modification can be performed in the second cell. For instance:
      1. **Want\_overlays** — Determines whether bead identifiers will overlay the images.
      2. **overlay\_type** — Selects the background type for the bead identifier annotations. You can switch between using “beads” and “droplets” as the background for the bead identification annotation.
      3. **Min\_beads** — Specifies the minimum number of replicates for a bead population to be included in the heatmap. Currently set to 3. If a bead population has 3 replicates, it will be shown on the heat map. If it is not, it will be a white square.
      4. **control\_classes** — Selects which controls are used for threshold calculations. Typically ["Scrambled 1","Scrambled 2"], but just [“Scrambled 1”] appears to also work.
      5. **mult\_outlier** — Sets the multiplier for outlier detection. Typically 1.5.
      6. **mult\_dev** — Determines the multiplier for standard deviation-based thresholds. Currently set to 3 but can be increased for a more stringent threshold.
      7. **fold\_minimum** — Minimum fold over the control class median donut intensity for determining thresholds. (e.g., 2 = threshold must be at least 2x median donut blue intensity for the control bead classes)
   3. Execute cells up to the eighth, which invokes a widget for optimizing cluster identification parameters.
      1. The way the widget works is that you can select a bead population and click on a place in the plot and it’ll update all the values. You can pick and choose the best clustering values for a given experiment.
      2. The radius value can also be changed from 0.15 to 0.12.
   4. Upon finalizing the clustering parameters, execute all subsequent cells.
   5. The analysis is now complete.

* **Additional Notes for Rerunning Analyses**
  + Should you wish to conduct a comparative analysis between multiple runs, it is imperative to segregate the output files from the initial run to prevent inadvertent overwriting.