Appendix 1:

Introduction

EC phenotyping relies on the quantification of parameters such as cell morphology, protein distribution and expression as well as junction heterogeneity, which can be measured by high-content image analysis techniques(Caicedo et al., 2017). Manual annotation of images to segment junctions can be prohibitively tedious and biased and are not suitable for large amount of data. To improve segmentation and analyse large datasets, specific software packages are available enabling also the analysis of features as complex as EC junctions(Brezovjakova et al., 2019; Seebach et al., 2015). However, most studies have been conducted using HUVEC in culture and did not consider Notch activation, cytoskeleton status or organ-specific responses.

We present a workflow (ECPT) using open-source software (ImageJ(Schindelin et al., 2012), Cell Profiler(Carpenter et al., 2006), R Studio(https://www.R-project.org/) which expands and improves our previous work by including cell cycle, NOTCH activation status, new morphology descriptor and improved descriptors of junctional status. Key improvements of the new workflow also include 1) the ability to phenotype widely heterogeneous EC without user input 2) the reporting of single cell measurements and 3) the ability to perform correlative analysis between the different parameters (at single cell level).

Here we describe the different steps of the pipeline and the software used to analyse images as well as the parameters obtained computationally.

Open access software and codes used in this study can be accessed here: (https://github.com/exr98/HCA-uncovers-metastable-phenotypes-in-hEC-monolayers)

Image loading:

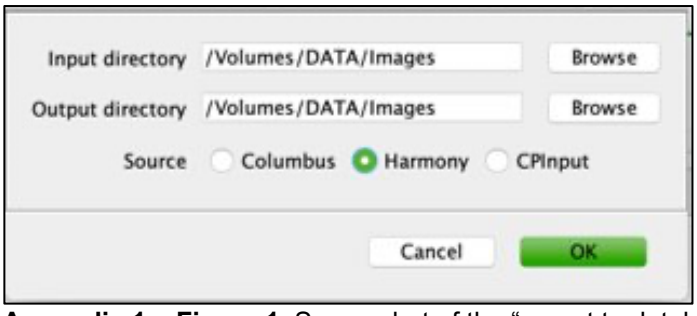
ECPT can be used on images obtained from different sources. In this study, we obtained raw images from an Operetta CLS system and exported images from Harmony.

The workflow can:

- handle different file format
- load and analyse multiple images at a time
- simultaneously analyse different signals from distinct channels (here, 4 different channels)

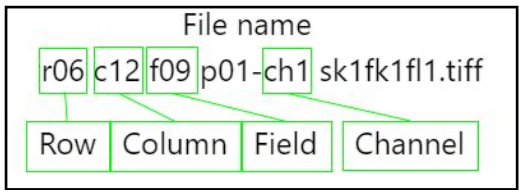
1) Harmony

Raw images sets were exported using the “export to database” function (Appendix 1- Figure 1) in Harmony generating a dataset composed of greyscale images (TIFF format, 2160x2160, 16 bits, one image for each channel).



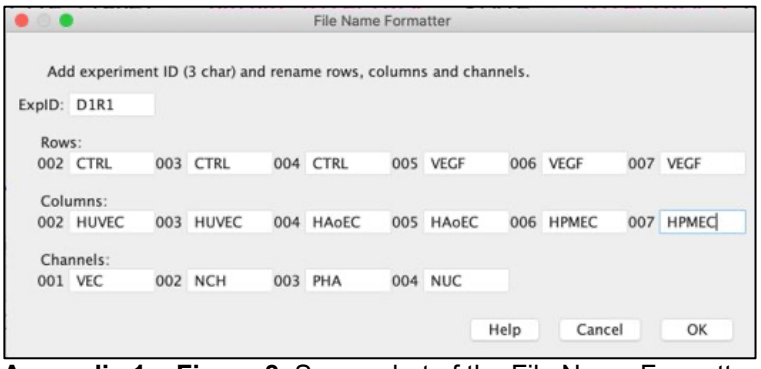
Appendix 1 – Figure 1: Screenshot of the “export to database” function.

The image names contain metadata (Appendix 1- Figure 2) which can be traced back to experimental conditions.



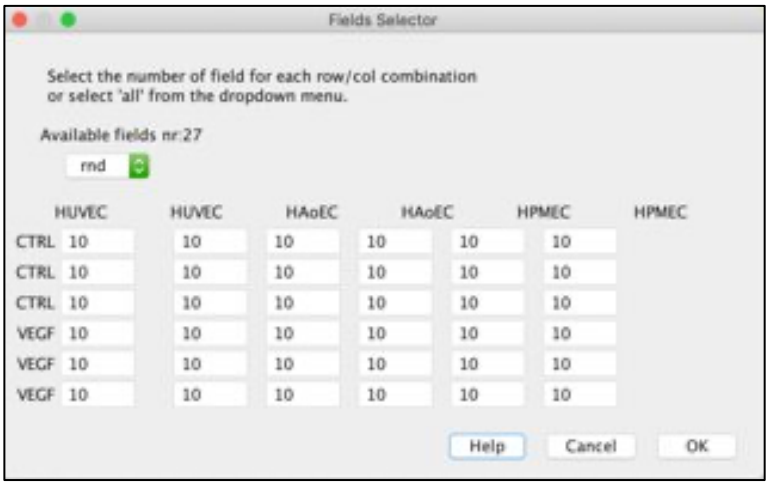
Appendix 1 – Figure 2: Example of the file names used in the experiment.

We developed an interface between Harmony (and Columbus) and Fiji (ImageJ) using these metadata. The Importer interface (Appendix 1- Figure 3) allows assigning desired values to row and columns metadata, in our exemplary experiment we assigned plate rows to cell type (HUVEC, HAoEC and HPMVEC) and plate columns to treatment (either CTRL or VEGF).



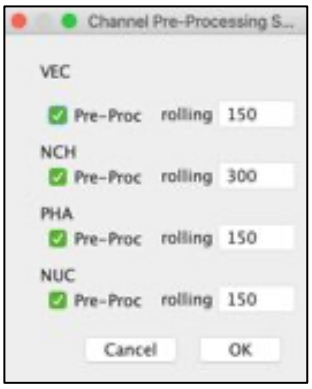
Appendix 1 – Figure 3: Screenshot of the File Name Formatter.

Once the row and wells names are defined, the interface allows selecting custom number of images (which can be randomly selected, or user defined) to be extracted from the database (Appendix 1- Figure 4).



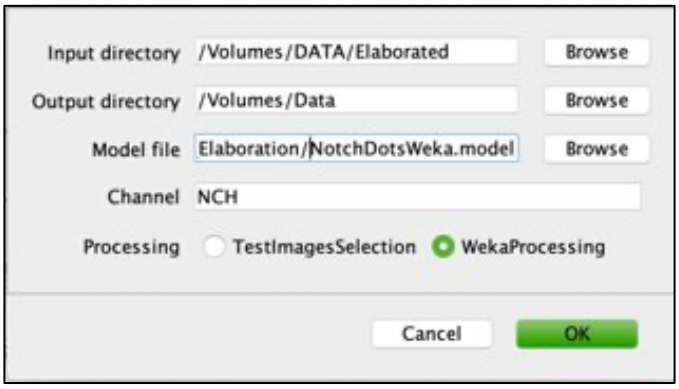
Appendix 1 – Figure 4: Screenshot of the Field Selector

Finally, the importer interface allows selecting image pre-processing options such as image re-scaling and illumination correction for each available channel (Appendix 1- Figure 5).



Appendix 1 – Figure 5: Screenshot of the Channel Pre-Processing selection window.

If these options are selected, imported images will undergo the selected pre-processing before being saved into a new database (Appendix 1- Figure 6).



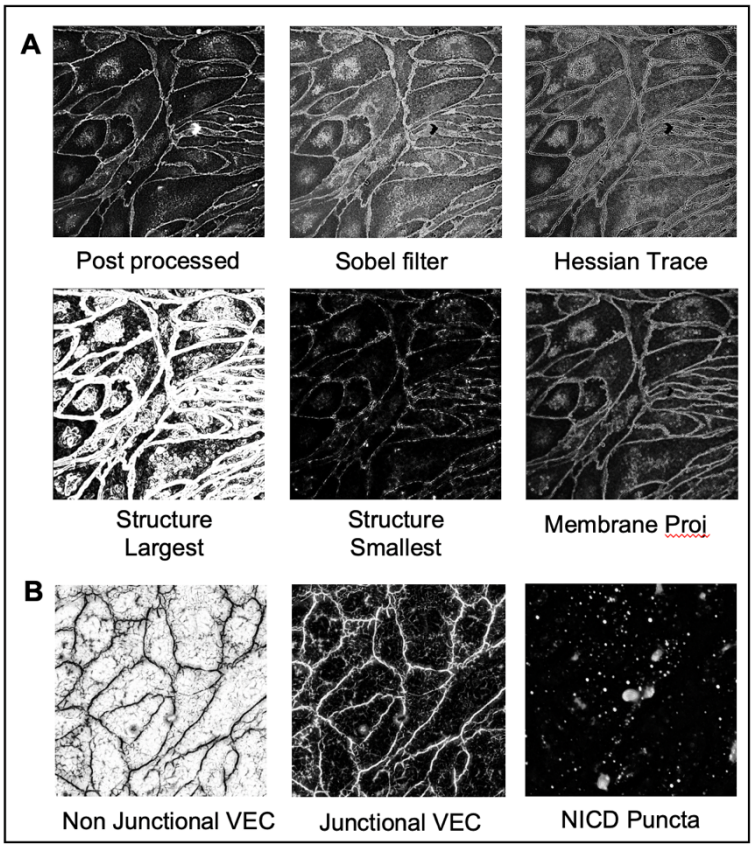
Appendix 1 – Figure 6: Screenshot of the Weka processing selection window.

The importer generates a new image set where new metadata are added to the file name and pre-processing is applied. In our example experiment we used the importer to assign Cell type and treatment to each image, we used pre-processing to rescale the images (2160x2160 px to 1080x1080 px) for lighter memory load in post-processing and to apply illumination correction. For illumination correction we used the subtract background function in ImageJ.

2) Fiji Weka segmentation and importer

The Trainable WEKA Segmentation (TWS), available as a FIJI (ImageJ) plugin (imagej.net/Trainable_Weka_Segmentation), is a powerful pixel-based segmentator. The TWS applies specific transformations on a given image and uses each transform as a layer for the trainer. The TWS offers many advantages in comparison to standard intensity threshold-based segmentation including the ability to discriminate texture features. We exploited these capabilities to distinguish junctional from cytoplasmic signal in CDH5 images which have similar intensity profiles but very different textures. The classifier separated images into three categories: cells (including cytoplasm and nucleus), junctions and overexposure.
For the segmentation of junctions, we empirically selected a minimal set of transforms including Gaussian Blur, Hessian, Membrane Projections, Anisotropic diffusion, Sobel filter structure and Neighbours, to speed-up processing (Appendix 1-Figure 7A). Hessian, Difference of Gaussian, and Membrane Projection filters were applied to aid in highlighting the junctions. The membrane thickness was set at 1, the membrane patch size at 19 and the sigma set from 1.0 to 16.0. The optimised classifier was then applied to the experimental images obtained.
Our feature stack consisted of 82 features (including the original image). To train the classifier to recognise junctional VEC in different EC types, we generated a stack of 70 images

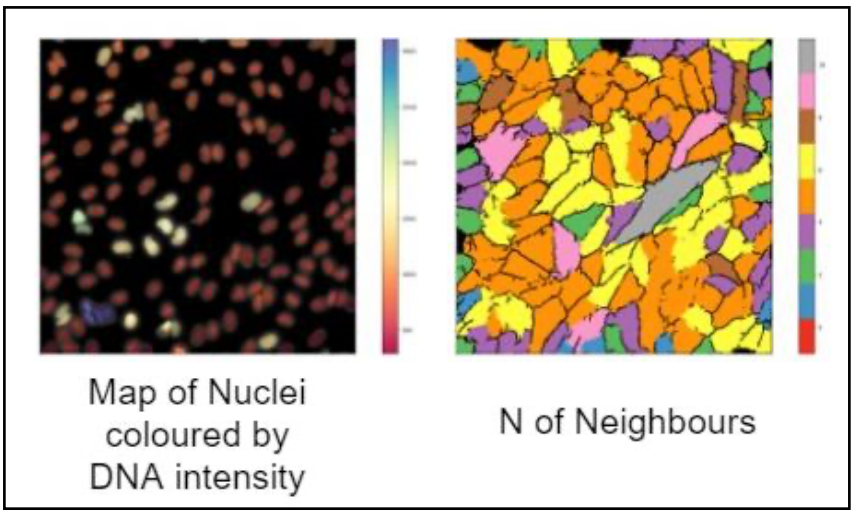
(200x200 pixels) extracted randomly (using an ImageJ macro) across our whole dataset. After training the dataset on this stack we applied the classifier model to all our original images (VEC channel after post processing) generating two new images (probability maps) for each original image (Appendix 1-Figure 7B, refer to TWS documentation for detailed instructions on training and applying models). A similar approach was used to segment puncta of activated NOTCH. The probability maps were used in downstream analysis (along with original post-processed images) with Cell Profiler.



Appendix 1 – Figure 7: Screenshots of images after processing. A) Original post-processed image (after illumination correction and example transform images with corresponding filter). B) Probability maps after application of WEKA models.

3) Cell Profiler pipeline

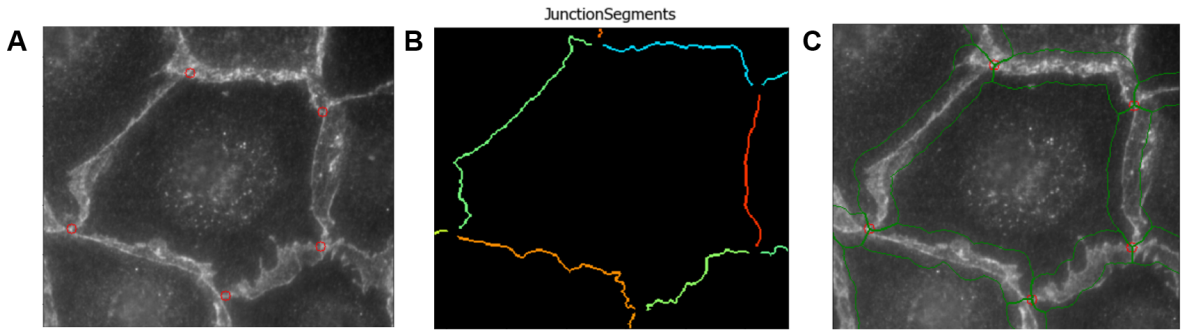
We designed a 60-module pipeline for EC characterisation to run on Cell Profiler (available at https://github.com/exr98/HCA-uncovers-metastable-phenotypes-in-hEC-monolayers). Cell Profiler[56] (CP) performs object-based segmentation based on different intensity thresholding strategies and it then allows manipulating and measuring these objects. A brief annotation has been made describing what we attempted to achieve with each module in the pipeline.
We used CP capabilities to identify the cell Nuclei based on Hoechst staining (Appendix 1-Figure 8, Map of nuclei coloured by DNA intensity), to identify cell body, cell junctions and NOTCH puncta using the probability maps generated by the TWS (Appendix 1-Figure 7B). After identifying each object, we made relevant measurements of intensity, granularity and colocalization of relevant fluorescence signals on the different objects.
From each nucleus, a corresponding cell object was then identified (IdentifySecondaryObjects #16). Mapping of the cytoplasm was obtained by removing the nuclei map from the obtained cell map (IdentifyTertiaryObjects #23). The cells' localisation and co-localisation were identified and measured, identifying the number of neighbours of each cell (Appendix 1- Figure 8, Number of Neighbours).



Appendix 1 – Figure 8: Cell Profiler pipeline outputs. The 47-modules pipeline for EC characterization was used to segment nuclei using DAPI staining, enabling mapping of Nuclei by DNA intensity.

Cell junctions were split into separate segments with each junction segment being shared by two adjacent cells. An outline of the cells (OverlayOutlines #25) was skeletonised which allows us to apply further morphological operations (MorphologicalSkeleton #26). The branch points were then obtained which corresponds to the ends of each junction segment (Morph #27). This was used to cleanly split the cell outlines into individual junction segments (Appendix 1-Figure 9). The junction segments were subsequently identified as objects (IdentifyPrimaryObjects #32) and various measurements for texture, granularity, intensity, shape and size were taken for downstream classification within CellProfiler Analyst. Feature selection was performed to select the most useful measurements within CP to help distinguish the different classes of inter-endothelial junctions (see the respective CP module annotations for elaboration on the rationale for the settings for each module). Once the junction segments have been identified, relationships are established to link them to the two respective endothelial cells that share the junction (MeasureObjectNeighbors #35). See the CellProflier Analyst section below for more information about how the junctions were classified by their phenotypes.

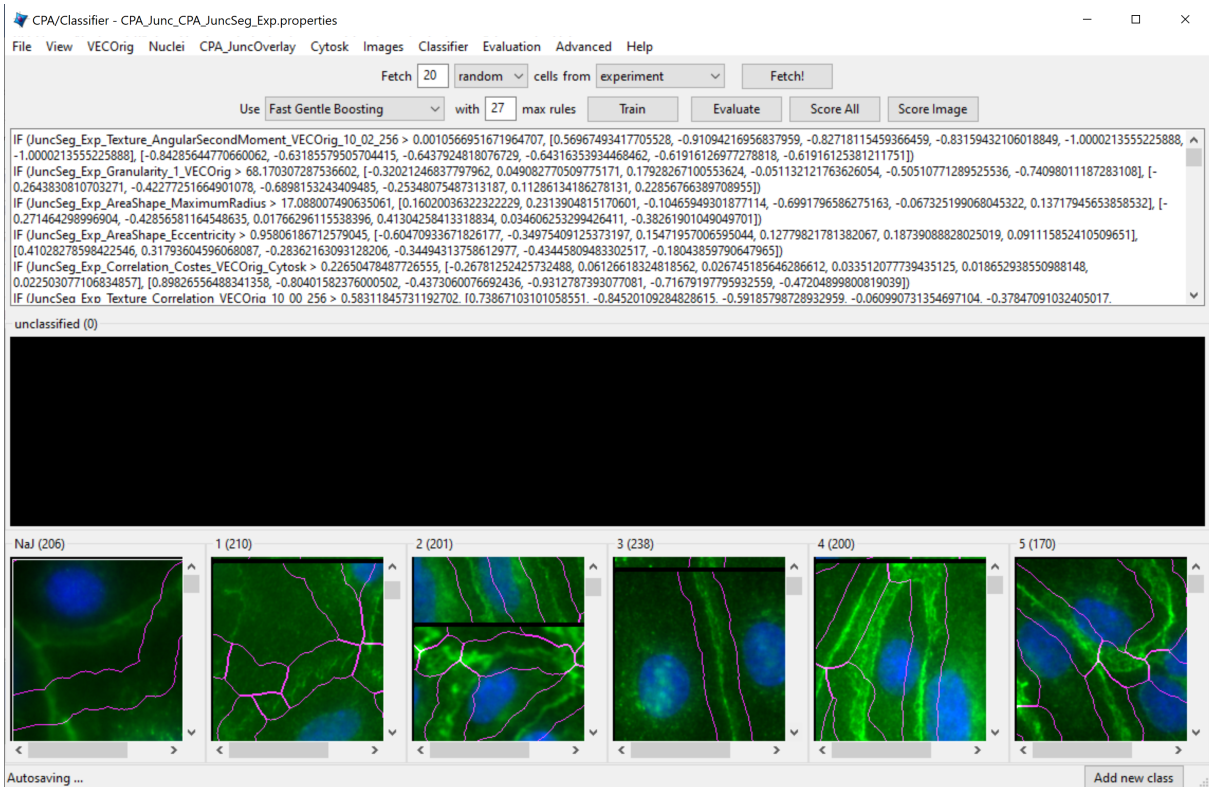
Total intensity of nuclear and cytoplasmic NOTCH and Hes1 was also measured independently (MeasureObjectIntensity #63). Previous experiments included the use of other stains such as phalloidin for actin filaments, showing the modular and customisable nature of the ECPT to different experiments.



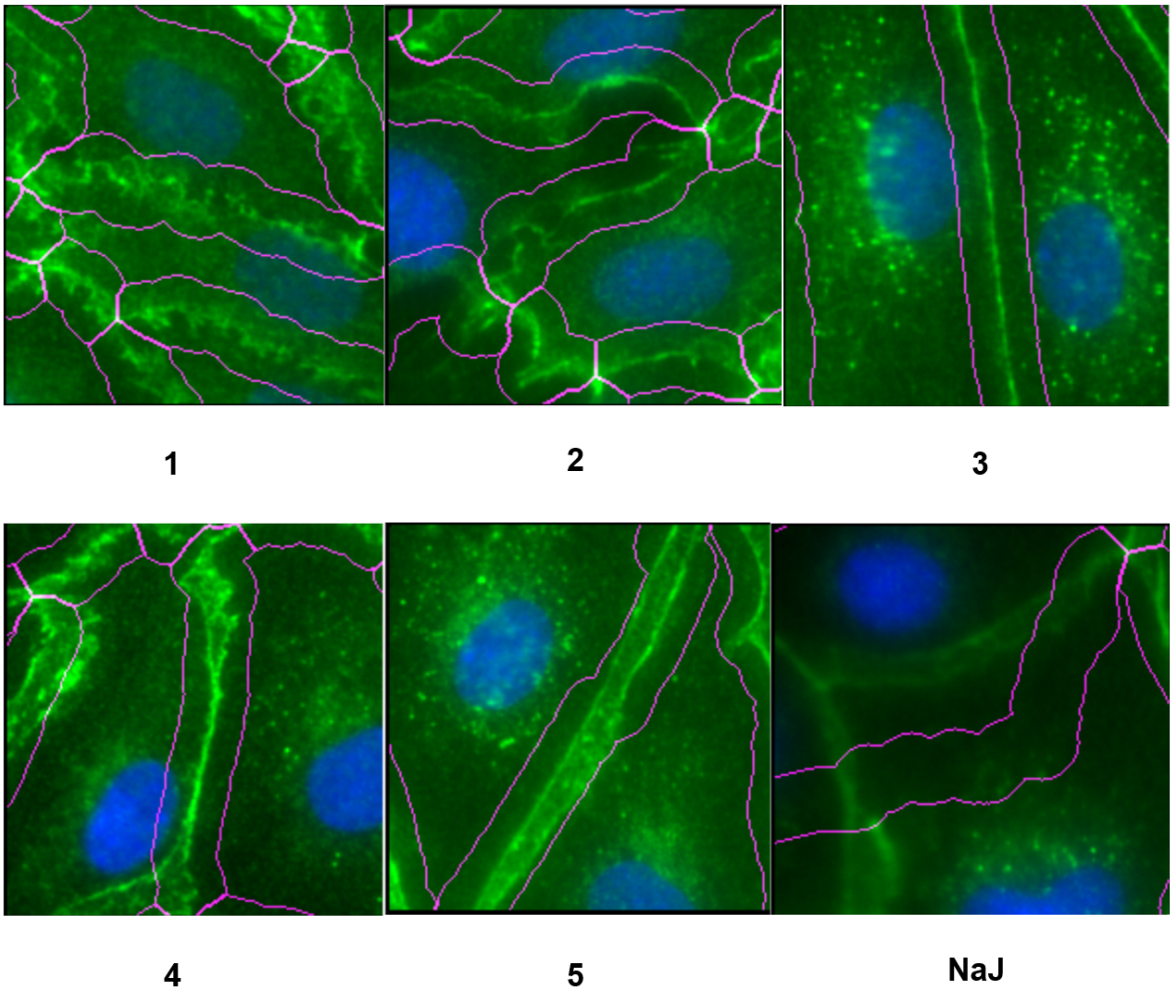
Appendix 1 – Figure 9 A-C: (A) Example of a cell with its branch points identified. (B) Segments of the example cell's junctions identified as individual objects. (C) An overlay of the identified junction segment objects to be classified in CellProfiler Analyst.

4a) CellProfiler Analyst (CPA) classifier

The CellProfiler Analyst Classifier model(Jones et al., 2008) was used to classify the different types of junctions with supervised machine learning. As described above, measurements were taken of the junction segment objects in the CellProfiler pipeline. These measurements are exported to an SQLite database for use in CPA. Junction segments were classified and trained using the Fast Gentle Boosting algorithm with 27 maximum rules (Appendix 1- Figure 10). The training set was manually created, sorting the different objects into 6 categories—classes 1 to 5 and "Not a Junction" (NaJ) (Appendix 1- Figure 11).

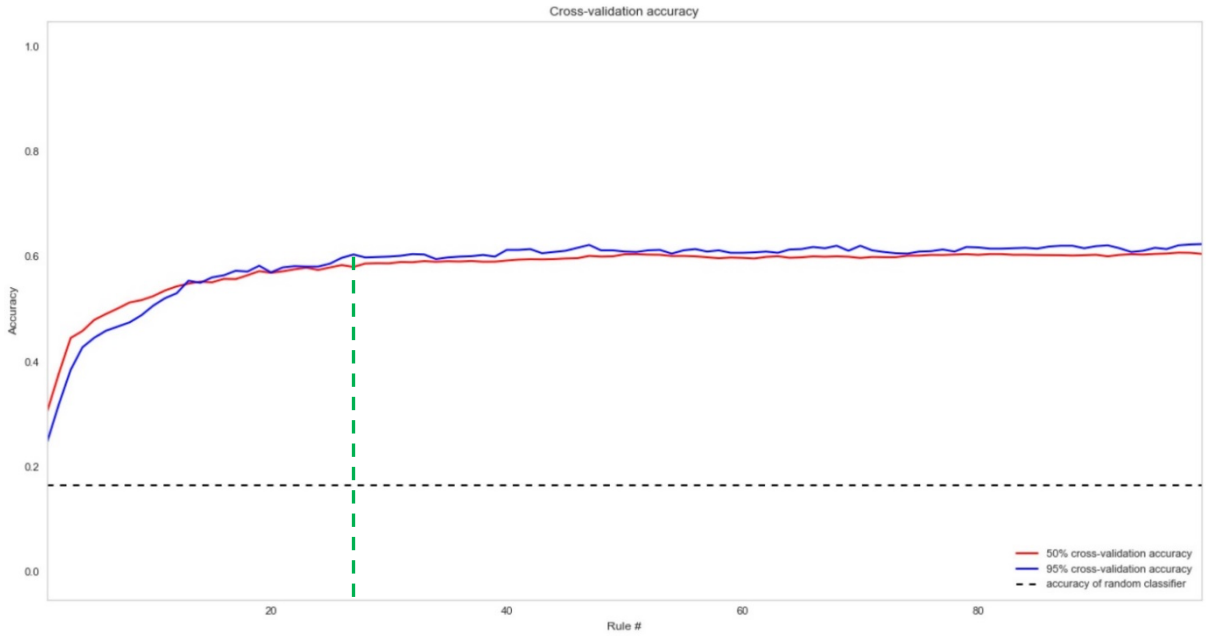


Appendix 1 – Figure 10: Screenshot of the Cell Profiler Analyst Classifier model. Images from junction area were manually classified into 6 different categories (Class 1-5 and NaJ).



Appendix 1 – Figure 11: Classes 1 to 5 and an extra class for objects that are not junctions (NaJ). Class 1 junctions are very jagged with many holes. Class 2 junctions are slightly jagged and may not be entirely continuous. Class 3 junctions are linear and continuous. Class 4 junctions often have a linear component with areas that have started to form reticulations. Class 5 junctions have overlaps throughout the junction appearing as double lines. NaJs are usually improperly segmented junctions or immunostaining artefacts.

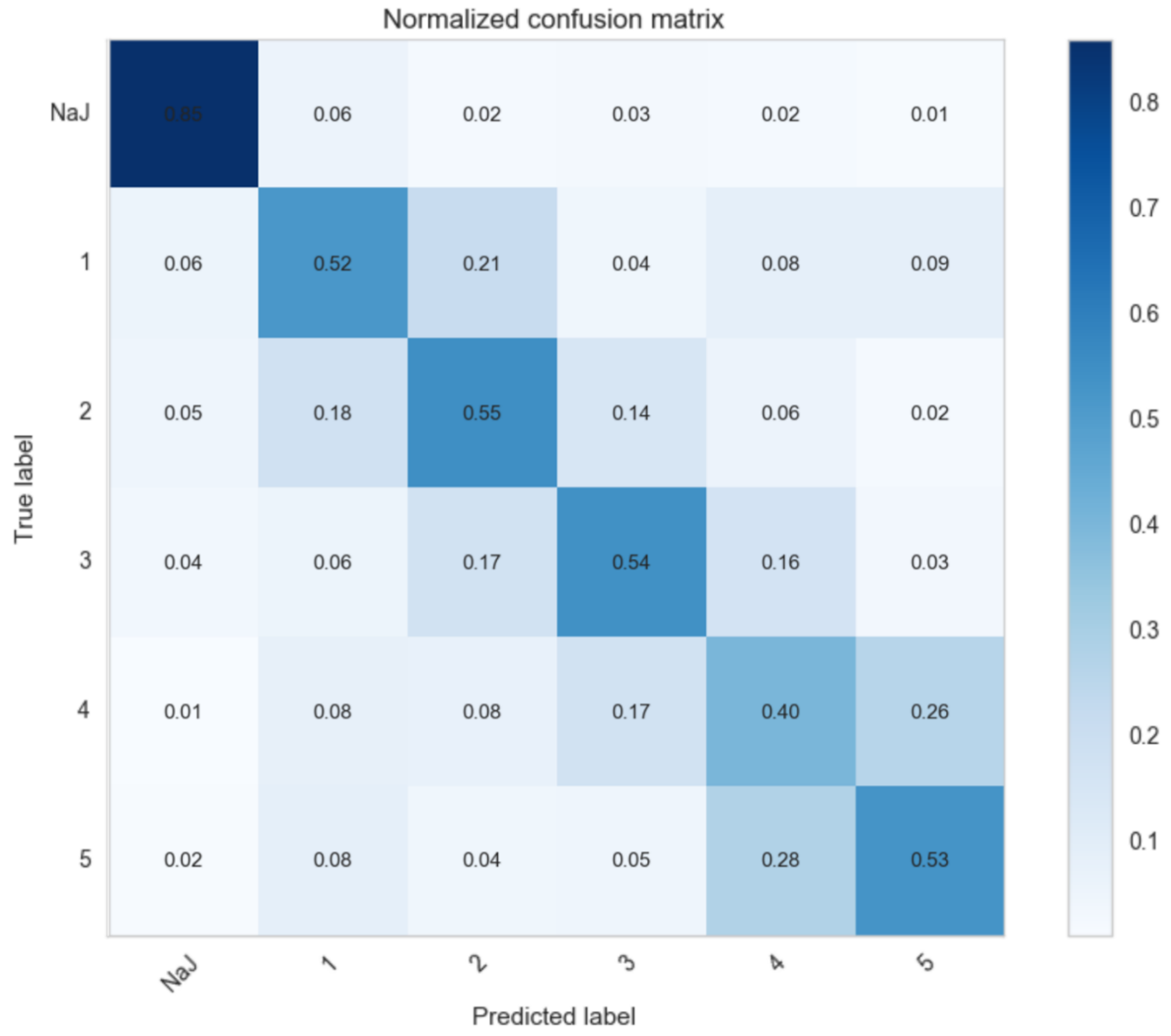
Each class was manually trained until around 200 objects were classified per class. The model was evaluated in CPA with a classification report (Appendix 1- Figure 12). This report shows an increase in accuracy as the number of rules increases. However, it quickly reaches a plateau and there are only marginal benefits for using the maximum number of 99 rules in CPA. Therefore, 27 rules were selected to reduce the computational requirements and prevent overfitting of the model to the training set.



Appendix 1 – Figure 12: Classification report of the model showing an increase in accuracy with an increase in the number of rules. In order to reduce computational requirements, a smaller number of 27 rules (green dashed line) would still result in an accurate model as compared to using a maximum of 99 rules.

The set of 27 rules were fed back into the CP pipeline to classify each junction segment object (FilterObjects #40-45). Further modules were added to the CP pipeline to measure the percentage of each junction class of the total cell's junction (MeasureObjectNeighbors #52-57). The length of each junction segment is also measured by measuring area of the single-pixel width skeletonised junction segments (MeasureObjectSizeShape #58). This information is again extracted to an SQLite database through the ExportToDatabase #84 module for downstream Moran's analysis in R.

The classifier was evaluated with a confusion matrix generated in CPA (Appendix 1 – Figure 13). NaJ objects were identified with an 85% true positive rate. Due to the similarity in appearance of certain junction classes, some classes are often mistaken for another class. For example, some class 2 junctions appear similar to class 1 and class 3. Class 4 junctions may have a section appearing linear like class 3 and another section appearing reticulated like class 5. Hence, a certain degree of misclassification has to be accepted as human assessors cannot consistently agree on the junction classifications.



Appendix 1 – Figure 13: Confusion matrix of the classifier model generated by CPA.

The model was compared against four human assessors involved in endothelial cell research at various levels of experience. 6 images and 10 well-segmented junctions per image were selected totalling 60 junctions used in the validation. The images were sent out to the assessors to be classified independently. The results of the human assessors were compiled and compared to four different classifier models trained with the exact same set of images. Even though the same junctions are used to train the classifier, each model produced can have slightly different parameters. This simulated four different classifier models versus four different human assessors.

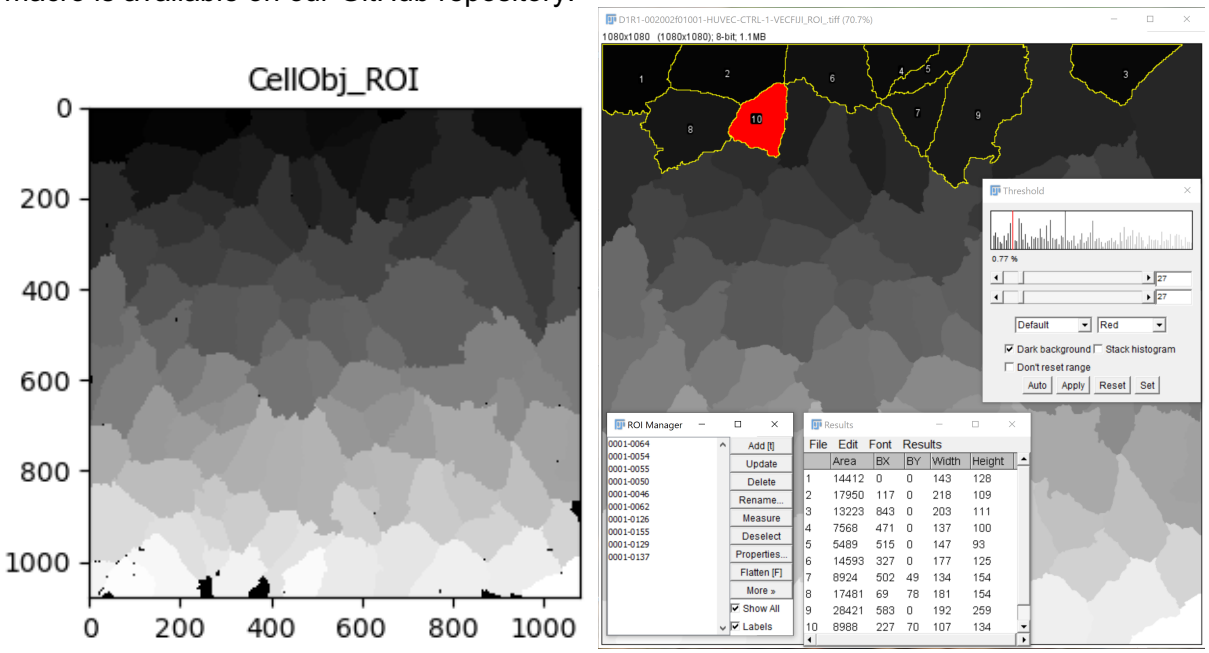
Unsurprisingly, humans showed more variation and inconsistencies in their classification while CPA models produced very similar results each time. Humans had an average standard deviation (SD) of ± 0.687 classes while CPA models had an average SD of ± 0.213 classes.

Manual assessment is strongly subject to observation biases which cannot be eliminated and requires a time-consuming session to reach a consensus classification. In contrast, automatic assessment is only subjected to systematic errors due to imperfection in the training of ML classifier. These errors can be recognised and eliminated either by improving the classification model or by compensating during downstream data processing.
Manual assessment of 60 junctions including drafting blind scores and agreed classifications took 5 h, therefore accurate manual assessment seems prohibitive even for less than 1000

cells (Using ECPT we assessed >300000 cells and evaluated >7.5 M individual junction objects).

5) Exporting Cell Regions of Interests (ROIs) to R for Spatial Analysis

This section describes the method used to establish a bridge between CP and R studio to perform spatial analysis in R using the data collected from CellProfiler. We employed a method to convert the cell areas as images into ImageJ ROIs. Endothelial cells segmented as objects in CP are converted to an image (ConvertObjectsToImages #20). This produces a grayscale image where the object number corresponds to its pixel intensity (Appendix 1- Figure 14A). An ImageJ macro was then applied to these images, thresholding at each pixel intensity to obtain a list of ROIs with the Analyze Particles function and the RoiManager (Appendix 1- Figure 14B). The output of which can be used in R with the RImageROI package. The ImageJ macro is available on our GitHub repository.



Appendix 1 – Figure 14: (A) Grayscale image from CP where each cell object number corresponds to its pixel intensity. (B) Example image showing 10 cell ROIs being processed in ImageJ. An ImageJ macro automates this process.

5) R Analysis

We employed R studio (https://www.R-project.org/) for all data analyses and representation. We connected R studio to the Master database using RSQLite package and imported measures of interest (Supplementary table 1). After importing we cleaned the result table to exclude mis-segmented cells and to rectify NaNs measures. The libraries and code snippets used for Database import, data normalisation, data clean-up, statistical analyses and to draw each plot are provided as R Studio Notebook in (https://github.com/exr98/HCA-uncovers-metastable-phenotypes-in-hEC-monolayers).