Imaging Mass Cytometry and Machine Learning: Cell Segmentation in Python

BMED320 Lab report, Fall 2020

Ajia Pennavaria[a,b]

[a] *Neural Networks Group (Lundervold lab), Department of Biomedicine, University of Bergen, Norway*
[b] *Mohn Medical Imaging and Visualization Centre (MMIV), Haukeland University Hospital, Norway*

Abstract

The simultaneous advancements in biomedical technologies and computer science have come together in image processing and tissue analysis through imaging mass cytometry (IMC). The aim of this project is two-fold: (**i**) Present the basic principles of IMC, and (**ii**) Demonstrate how we can read, display and perform initial analysis of IMC data from scratch in Python using a public and openly accessible dataset related to the work by N. Damond et al. "A Map of Human Type 1 Diabetes Progression by Imaging Mass Cytometry". Cell Metabolism 2019;29(3):755-768, and Jupyter notebooks (`https://jupyter.org`). Being able to analyze intact tissue samples at sub-cellular resolution, spatial organization and cellular phenotypic characteristics can be revealed which help aid in the understanding of complex tissue function. Though such information requires machine learning methods to accurately visualize and interpret the data. In IMC, single cell segmentation represents one of the most important preprocessing steps where several supervised machine learning software have been developed. Nonetheless, histogram thresholding and unsupervised K-means clustering, accessible through easily imported Python libraries, show the capability to perform cell segmentation, manipulating the high-dimensional dataset into easily readable formats. Although these unsupervised and from scratch methods may not operate on par with more robust computational tools, they do illustrate the biological applications of fairly simple and common machine learning algorithms.

Keywords: Imaging Mass Cytometry (IMC), Cell segmentation, Machine Learning, Python, Otsu's method, K-means clustering, Jupyter Notebooks

PART 1: Theory

1. Underlying technologies and principles of imaging mass cytometry

Understanding the interplay between tissue function and disease is critical in the future developments of diagnostics and therapeutic treatments. The fundamental technologies of flow cytometry and mass spectrometry have contributed to the emergence and advancement of mass cytometry and its subsequent expansion, imaging mass cytometry. Mass cytometry, also known as cytometry by time-of-flight (CyTOF), brings together these two technologies to analyze single-cell suspensions using heavy-metal isotope labeled antibodies [1], instead of fluorescent molecules. This distinct difference solves the issue of spectral overlap when analyzing samples with several fluorescent reporters simultaneously, which limit the number of detectable parameters in flow cytometry [2]. In

Email address: `Ajia.Pennavaria@student.uib.no` (Ajia Pennavaria)

contrast, mass cytometry utilizes separation by mass spectrometry, which results in a significantly improved discrimination between reporters of different atomic weights with a high degree of accuracy [1]. Reporters molecules, often antibodies, are directed at specific biomarkers of interest and measured through the process of labeling to heavy-metals primarily from the lanthanide series [3]. Currently, availability limitations of these elemental reporters restrict the number of detectable cellular biomarkers to less than 50 [4], but provide a strong basis for antibody panels including but not limited to tumor, immunological, and pharmacological investigations [5, 6]. However, as the preservation of tissue architecture remains the gold standard in histopathology when diagnosing a number of diseases [7], it is subsequently lost in both flow and mass cytometry as the cells must be suspended in a buffered salt-based solution [8, 9].

While flow cytometry and mass cytometry are well established options for rapid high throughput analysis of tissue composition and protein expression; spatial location and cellular interactions are lost in the suspension. Imaging mass cytometry (IMC) offers a solution of unparalleled technology, utilizing non-optical imaging techniques, supplying researchers with the means to obtain a wealth of biological information. It provides the ability to distinguish tissue specific cell types, the simultaneous measurement of the expression of up to 40 proteins, study cellular interactions within their native microenvironment, and spatial analysis in intact tissue samples [10]. The instrumentation used to achieve this is the Hyperion^{TM} imaging system from Fluidigm which consists of a two-part laser ablation system and mass cytometer [11]. Histological tissue sections are first treated with panels of antibodies, each labeled with a unique heavy-metal isotope directed at a specific biomarker or protein. The tissue slide is inserted into the laser ablation chamber and the regions of interest (ROI) are scanned by a highly focused pulsed UV-laser at 213 nm. As the slide moves under the fixed laser beam, spots of 1 μm in diameter are ablated and the tissue vaporized [3]. The plume of particles are carried into the inductively coupled plasma ion source, atomized, and simultaneously quantified by TOF mass spectrometry [12]. Measurements of the isotopes at each ablation spot corresponds to a specific protein within the tissue and represents one pixel in the resulting image. This sequential process is illustrated in Fig. 1.

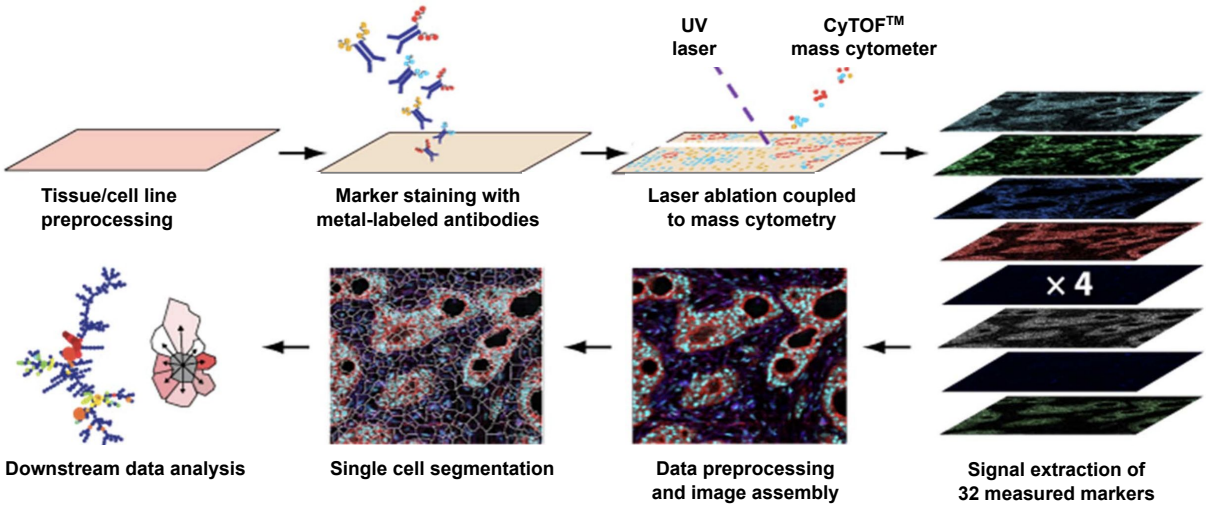

Figure 1: Simplified overview of imaging and cell segmentation in IMC. Histological tissue sections are stained with panels of heavy-metal conjugated antibodies and subjected to laser ablation. Measurements at each ablation spot of each isotope are quantified using TOF mass spectrometry and image stacks of the tissue are generated. Image and data preprocessing is performed and applied to cell segmentation software which allows for further analysis at sub-cellular resolution (from Jørn Skavland's BMED320 lecture, UiB).

The process of spot-by-spot ablation over the ROI, generates a stack of gray-scale images depicting individually measured protein expression and anatomical detail. However, the data generated requires specialized software to provide accurate visualization and interpretation of the images. [10, 3] (cf. example in Fig. 2).

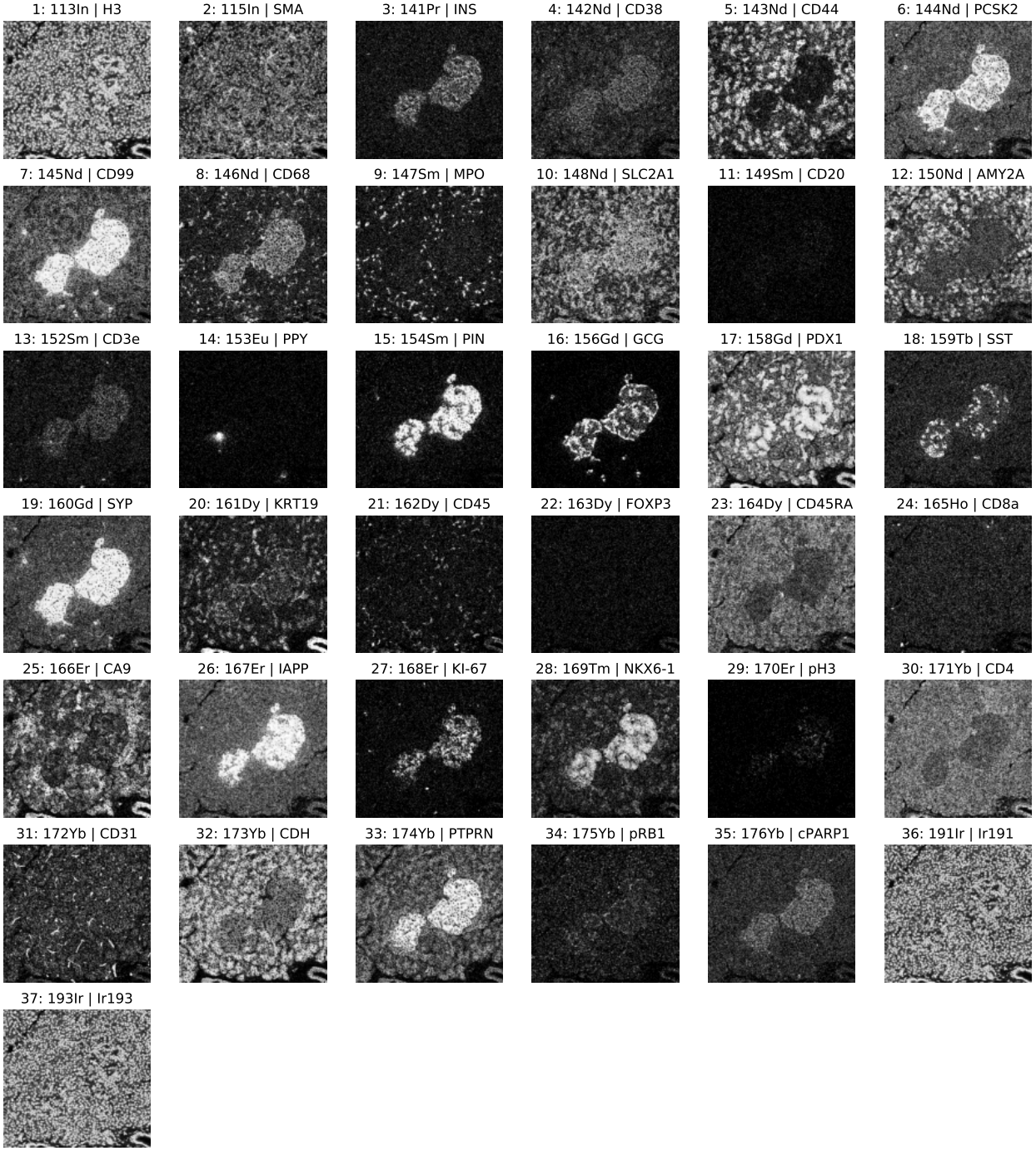

Figure 2: Mosaic of histogram equalized channel images in the E08 IMC dataset [13] used in the experimental part of this project, where the metal tag and short name for its antigen is denoted above each of the 37 channels (figure produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-1.ipynb`).

2. Computational imaging and machine learning in IMC

The accumulation and acceleration by which biomedical data is being produced directly coalesces with the advancement of new technological methods and computer science. Unprecedented amounts of data can be generated, which pose significant challenges when trying to manage and interpret the data in an effective way [14]. One method that has been implemented in a number of fields to solve this issue is machine learning, an application of artificial intelligence that utilizes carefully constructed algorithms to better analyze large datasets and improve many of the ways we interact with modern technology [15]. For instance, computational imaging, essentially a lens-less system uses machine learning techniques to predict, classify objects, and reveal significant patterns from the raw data input [16] that may have otherwise been missed due to the limitations of human perception [17]. In IMC, the non-optical system generates images characterized by protein expression in tissue sections at a resolution of 1 μm per pixel. The images generated represent a highly complex system of cellular phenotypes and spatial organization which require computational methods to analyze and interpret [10]. It is beyond the scope of this article to present all possible computational methods used in image processing but will instead provide a brief overview of the steps with a strong focus on methods for single-cell segmentation. A short description of the differences between supervised and unsupervised machine learning will also be discussed at the end of this section.

Before raw IMC data can be analyzed by various machine learning methods, it needs to be converted into a usable format for further downstream analysis. This is done by converting the files into tiff images and can be achieved using the preproccessing script supplied by Bodenmiller Group [18]. Pseudo color images can then be reconstructed where the colors assigned correspond to specific proteins within the tissue. By stacking multiple images that depict several biomarkers simultaneously, it is possible to visualize protein distribution throughout the ROI [3].

In order to extract the information obtained through IMC analysis, cells contained within the generated images need to be segmented, and is arguably the most important and challenging step in IMC data processing [19]. This process enables further phenotypic characterization at the single-cell level as well as quantification of cell frequency in a heterogenous population and spatial analysis [20]. One program is the `Ilastik` segmentation tool kit which enables pixel classification of an image based on the probability that each pixel belongs to one of multiple classes defined by the user [21]. This machine learning software can then be used in conjunction with `CellProfiler`, another machine learning tool available as an open source image analysis software. The classification masks obtained by `Ilastik` are uploaded into `CellProfiler` which works to identify and outline cell edges or cell interiors by identifying primary and secondary objects within the image. It does this by measuring cellular features such as cell count, size, shape, texture, and signal intensity [19].

Another widely used approach is that of segmentation by clustering. K-means clustering is one of many methods and is among the simplest unsupervised algorithms to divide large datasets into K amount of clusters [22]. The goal is to produce clusters with high intra-class similarity and low inter-class similarity, meaning that objects within the clusters are more similar to one another than objects in other clusters [23]. When working within a specific ROI, pixel features are extracted and then grouped together. In IMC, CyTOF metal-isotope signatures function as the features and are clustered based on their distance from one another. Because the number of clusters must be assigned manually, a different number of clusters will produce different results. It follows then that this parameter must be chosen carefully as to achieve the desired segmentation [22].

An even simpler method, and one to our knowledge that is not as frequently used on it's own in IMC image analysis, is single-cell segmentation by way of thresholding. This technique has however been used for the segmentation process of nucleus labeled fluorescent images in *C. elegans* [24], as well as RNA interference fluorescent screening of *Drosophila* [25]; both of which clearly demonstrate

that cell segmentation can be achieved in a basic way without the use of specifically designed machine learning software. This approach requires the user to define the dataset, only selecting channels that correspond to DNA markers and returns a pseudo-color image based on pixel intensity. This is then applied to thresholding algorithms, calculating a threshold value to further separate the image into the foreground and background [26]. Although intensity thresholding has been used since the 1960's and may be implemented in more sophisticated segmentation programs, alone it can produce relatively poor results [27].

Once cells within the image are segmented it is possible to quantify single-cell marker expression and visualize cell identification labels. With programming packages such as `histoCAT`, neighborhood analysis can be performed which group individual cells into sub-populations based on spatial features that enables the study of cell-cell interactions. Heatmaps are generated which depict the significance of interactions between or within cell phenotypes and allow for the study of cellular networks within the microenvironments of the tissue [28].

The concept of machine learning has been mentioned several times throughout this section but still may be an unfamiliar topic to some of the readers. As is described here [29], the goal of machine learning is essentially to develop mathematical models that enables computers to process input data and perform tasks such as data sorting, pattern recognition, classification, and even make predictions on new data. The terms supervised and unsupervised refer to how the model is built up and trained in order to execute specific tasks. In supervised machine learning a computer is provided training examples of correctly labeled data and user defined categories, whereas unsupervised machine learning is not [30]. The software used in IMC image processing is no exception. Both `Ilastik` and `CellProfiler` are supervised methods, learning from training data while continually optimizing the algorithms for feature classification and cell segmentation [21, 31]. Clustering algorithms on the other hand, are unsupervised methods as they are used to classify datasets without any prior knowledge or training [32], but learn from the unlabeled data itself [33].

PART 2: Computational experiments - "IMC from scratch: in Python"

3. Introduction

We have based our computational experiments on the IMC paper by N. Damond et al. [13] and corresponding open data provided by the authors on `Mendeley`, i.e. `https://data.mendeley.com/datasets/cydmwsfztj/2` (version 2, 09-04-2020). More specifically, we have been working in a Jupyter notebook using the dataset `E08_a0_full.tiff` and the corresponding metadata `E08_a0_full.csv` from case 6126 as is also illustrated in their original article [13]. We were able to access and use the published data to perform alternative methods of single-cell segmentation using DNA channel signals combined with histogram-based thresholding and K-means clustering. While those that are interested in the entirety of the study in disease progression of T1D are encouraged to read the original article [13], a very brief description will be given here. Damond et al. obtained histological pancreas sections from 12 human donors that were categorically organized according to disease duration and analyzed by IMC. Islet-cell composition at the head and tail of the pancreas were selected as the ROI in order to study β-cell destruction throughout the stages of T1D. The selected data included image stacks across all channels as well as panel information on the antibodies and their unique metal tags. Specifically, image E08 from case 6126, a non-diabetic donor whose sample was taken from the tail end of the pancreas is the data being used in this experiment.

To be able to analyze the experimental IMC data, researchers need more than just machine learning algorithms, they need a digital environment specifically adapted to meet their computational and programming needs. Jupyter notebook provides such an environment where users can interact with the data, and modify and repeat experiments all in one document [34], organizing code input and the computational output into chunks which can be run individually and modified accordingly [35]. By defining the dataset in the notebook, Jupyter allows us to read and print characteristics of the datafile. Working in the Python programming language, one can easily import libraries within a Jupyter notebook such as `NumPy`, `pandas`, `Matplotlib`, `seaborn`, and `scikit-learn`, allowing for a more in-depth interaction with the data. Using `NumPy`, images can be stored, sorted, and structured as arrays of numbers suitable for numerical calculations, while `pandas` is used to manipulate the data and attach labels to the sorted rows and columns. `Matplotlib`, along with its application programming interface `seaborn`, is used to plot data for visualization and statistical modeling in a variety of ways, and `scikit-learn` is a Python package with a built in number of machine learning algorithms implementing both supervised and unsupervised methods [36].

Working within the obtained dataset, it will be demonstrated how IMC data of all measured channels can be displayed in the computational environment of Jupyter notebook as well as how single-cell segmentation can be achieved using thresholding techniques and K-means clustering. A very moderate interpretation of downstream analysis will also be explored through correlation heatmaps. The goal is to illustrate IMC image analysis in Python where supervised software is typically preferred.

4. Methods

We first present the data being used, i.e. the panel of antibodies and corresponding channel images, as well as a description of the dataset before we describe image analysis and the machine learning procedures. It is important to mention that the segmentation methods used here were performed based on trial and error in accordance with the different techniques as described in section 2.3. Given that programming has an innately flexible nature, no exact protocol was followed.

4.1. IMC image data and panel metadata

Table 1: E08 antibody panel data depicting the antigen (biomarker) of interest, main target cells, metal tags and corresponding IMC channels (table produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-1.ipynb`).

PanelNo	Short Name	Antigen	Main Target Cells	Metal Tag	Chn
1	INS	Insulin	β	141Pr	3
2	PIN	Proinsulin	β	154Sm	15
3	GCG	Glucagon	α	156Gd	16
4	SST	Somatostatin	δ	159Tb	18
5	PPY	Pancreatic polypeptide	γ	153Eu	14
6	NKX6-1	Homeobox protein Nkx-6.1	β	169Tm	28
7	PDX1	Pancreatic and duodenal homeobox 1	β δ ductal	158Gd	17
8	IAPP	Amylin	β	167Er	26
9	PCSK2	Proprotein convertase 2	α	144Nd	6
10	SYP	Synaptophysin	Endocrine	160Gd	19
11	CD99	CD99	Endocrine	145Nd	7
12	SLC2A1	Glucose transporter 1	Endocrine	148Nd	10
13	PTPRN	Receptor-type tyrosine-protein phosphatase-like N	Endocrine	174Yb	33
14	AMY2A	Pancreatic amylase	Acinar	150Nd	12
15	KRT19	Cytokeratin 19	Ductal	161Dy	20
16	CD44	CD44	Exocrine	143Nd	5
17	CD45	CD45	Immune	162Dy	21
18	CD45RA	CD45RA	Immune	164Dy	23
19	CD3e	CD3	T	152Sm	13
20	CD4	CD4	Helper T	171Yb	30
21	CD8a	CD8a	Cytotoxic T	165Ho	24
22	CD20	CD20	B	149Sm	11
23	CD68	CD68	Monocytes macrophages	146Nd	8
24	MPO	Myeloperoxidase	Neutrophils	147Sm	9
25	FOXP3	Forkhead box P3	Regulatory T	163Dy	22
26	CD38	CD38	Immune	142Nd	4
27	CDH	E-/P-cadherin	Epithelial	173Yb	32
28	CD31	CD31	Endothelial	172Yb	31
29	SMA	Smooth muscle actin	Stromal	115In	2
30	KI-67	Ki-67	Proliferating	168Er	27
31	pH3	Phospho-histone H3	Proliferating	170Er	29
32	pRB1	Phospho-retinoblastoma	Cycling	175Yb	34
33	cPARP1	Cleaved caspase 3 + cleaved poly (ADP-ribose) ...	Apoptotic	176Yb	35
34	CA9	Carbonic anhydrase IX	Hypoxic	166Er	25
35	H3	Histone H3	Nuclei	113In	1
36	Ir191	(DNA intercalators)	DNA	191Ir	36
37	Ir193	(DNA intercalators)	DNA	193Ir	37

After reading the IMC data (provided as `TIFF` file) into `NumPy` arrays, we display the channels separately using pseudo-color encoding as shown in Fig. 3.

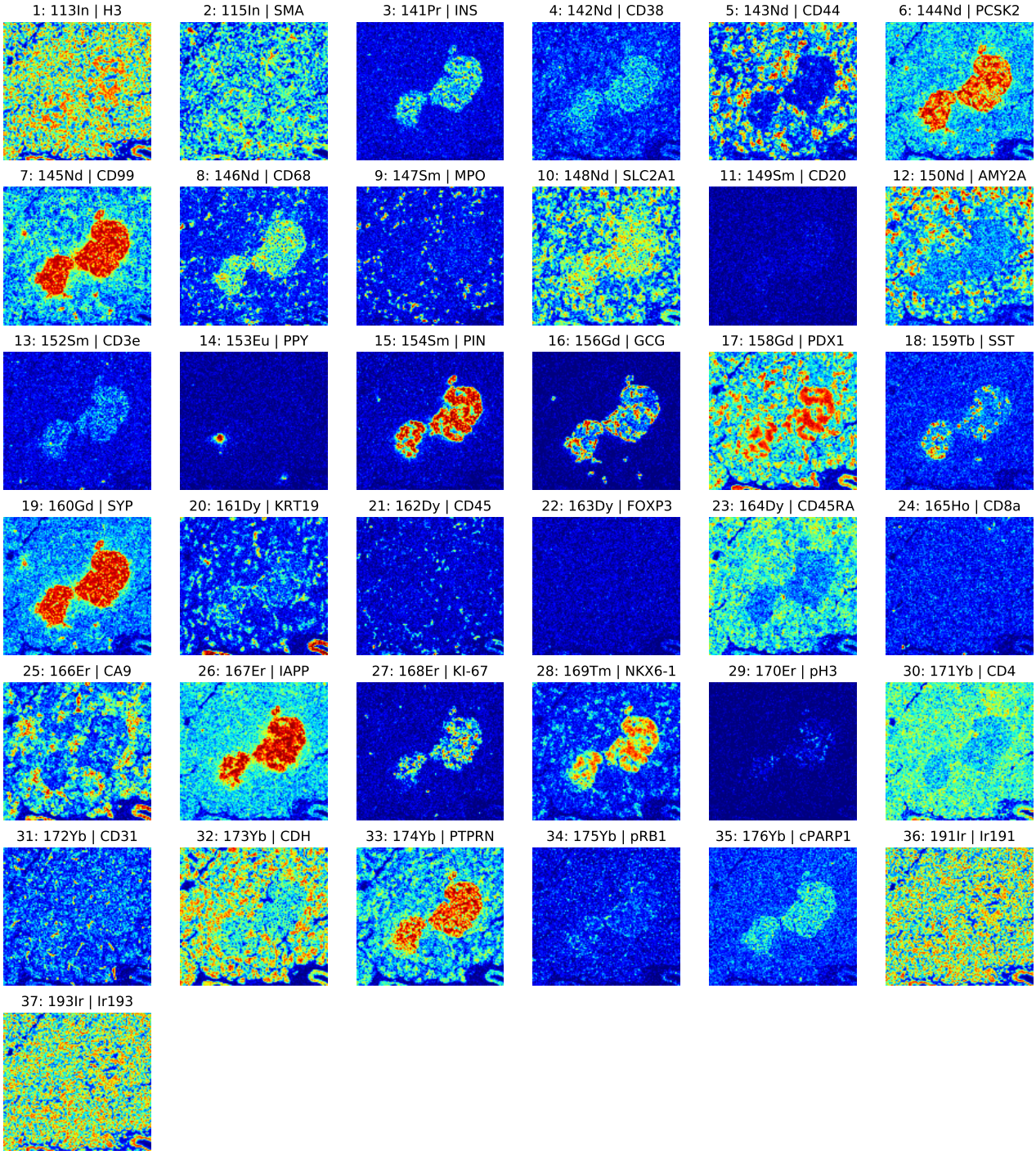

Figure 3: Mosaic of the 37 pseudo-color coded channel images in the E08 IMC dataset. Headings denote: `Chn: Metal Tag | Short Name` (figure produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-1.ipynb`).

Because the image stacks demonstrated a large range of values of pixel intensities, a dataframe of the dataset was extracted, scaled, and saved. This was done using the function `StandardScaler()` from `scikit-learn`, a form of normalization where all measured CyTOF pixel signatures were z-score transformed to have a mean of 0 and standard deviation of 1. As a result, the scaled pixel intensities were shifted, resulting in a majority of negative values. Although the original IMC data did not include negative measurements, the scaling function was performed so that each pixel feature contributed proportionally to the distance-based K-means algorithm and Pearson correlation.

4.2. Cell segmentation using DNA markers and thresholding

The datafile provided as `E08_a0_full.tiff` was used as input and loaded as `NumPy` arrays. Channels 36 (191Ir) and 37 (193Ir) were then selected for the cell segmentation process,, where each channel and corresponding data were preprocessed using histogram equalization and gray-scale colormap. Using `Matplotlib`, two images were produced that represent pixel intensity for both channels. To find the point-wise maximum of the two images we then ran the function `numpy.maximum()`, comparing the two channel arrays where the highest intensity value at each pixel location was used to return a new array and a third image. This third gray-scale image was then used as the input to calculate a threshold value based on Otsu's method as implemented from `scikit-image`, an imaging processing library in Python. As described here [37] Otsu's algorithm computes a bimodal histogram of two distinct peaks (not shown) for our point-wise maximum image and calculates an optimal threshold value to minimize intra-class variance between the distribution of black and white pixels. The algorithm assumes our 2D image consists of either the foreground or background to return a binary mask.

4.3. Choosing the appropriate number of clusters for K-means

No prior knowledge of the dataset indicated an appropriate number of clusters (K) to assign during segmentation by K-means and was therefore determined using two evaluation techniques, the elbow method and silhouette method. The K-means algorithm used here was implemented from `scikit-learn`.

The elbow method involves a visual determination of the resulting curve's sweet spot where K-means is run repeatedly on an increasing number of clusters for the dataset. Using the scaled dataset, K-means was run iteratively for $K \in \{1, \ldots, 10\}$ calculating the corresponding sum of squares error (SSE) as a performance indicator and plotted as a curve. The value of K where SSE dramatically decreases is known as the inflection point or sweet spot and represents the number of real clusters in the data [38]. The inflection point of the curve was deemed too difficult to determine by visual analysis alone and was programmatically identified using knee-point detection in Python. The `kneed` package, which can be found at the GitHub repository here [39] was used to find the maximum point of curvature to determine the number of clusters.

The silhouette method is used to assess the validity of clustering and relies on calculating a score based on the average distance between points within the same cluster and average distance between a point of one cluster to the nearest cluster that it is not a part of. Using the `scikit-learn` implementation of the silhouette method and scaled dataset, K-means was looped and run over an increasing number of values for $K \in \{2, \ldots, 10\}$, calculating the silhouette score for each assigned number of clusters and plotted as a graph. The resulting silhouette coefficient ranges from -1 to 1 where the values closest to 1 indicate good intra-cluster cohesion and effective separation [38].

4.4. Unsupervised classification of the IMC stack - K-means clustering

K-means clustering was performed in order to segment the dataset into different groups of similar features based on pixel intensity signatures. Pixel intensities, or CyTOF metal-isotope measurements from our 2D tiff images represent the features living in $\mathbf{R}^n$, where $n = 37$. Unsupervised

segmentation by K-means was performed over the scaled dataset using conventional K-means as implemented in `scikit-learn` and loaded as `NumPy` arrays. Using `pandas`, the scaled dataframe was read and displayed to show pixel intensities across all locations for each of the 37 measured channels. The K-means model was fit to the dataset for $K = 3$, with a max number of iterations set to 300, where K-means iteratively reassigns each point to its closest centroid and calculates the centroid mean assigned to each cluster. These two steps repeat until the assignments no longer change or the number of set iterations are met and the within-cluster variances minimized [38]. Each pixel was assigned it's cluster and used to reconstruct a pseudo-color red, green, and blue segmented image using `Matplotlib`. To illustrate the degrees to which image segmentation results differ when assigned a different number of clusters we also performed cell segmentation by K-means with varying values of K. The K-means algorithm was run and looped through the scaled dataset for $K \in \{2, \ldots, 10\}$ where clustering results were then generated using `Matplotlib`, given a grey-scale, and labeled with corresponding K-values.

4.5. Pearson correlation between CyTOF features pairs

In Python, `pandas` provides a function to find the correlation between variables within the created dataframe. That is, a function to and assess the statistical linear relationship between two numerical variables, ie. pixel intensity values across the image stacks. To do this, the Pearson correlation coefficient was calculated for the metal tag pairs across all 37 channels. After reading the scaled dataset into a new dataframe, the function `dataframe.corr(method='pearson')` was applied, specifying the function argument as the Pearson method of correlation. A visual representation of the calculated correlation values was generated using `seaborn` and displayed as a heatmap using a specified yellow-green-blue color gradient to illustrate differences in pairwise association, along with their numerical correlation values. The correlation matrix was generated as a symmetric plot where results could be interpreted from either the upper or lower triangle. Because of the redundancy in information a mask was used to hide one side of the matrix, displaying only the lower triangle. The results could then interpreted using the conventional correlation values described in this study [40].

5. Results

5.1. Cell nuclei segmentation using the two DNA channels and Otsu's binarization

In Fig. 4 , IMC channels 36 and 37 show a representation of pixel intensity values across the image where measurements of the heavy-metal isotopes correspond to pixel intensity at each location. By setting the colormap to gray, `Matplotlib`'s `imshow()` controls the gray color of each pixel where the minimum intensity value (ie. pixel intensity = 0.0) for the image is set to black and the maximum intensity value is set to a white color in the resulting gray-scale image. Because 191Ir and 193Ir are both used to target cellular DNA, the resulting images look very similar. Pixels that have a higher intensity value, and thereby higher measurements of the DNA targeting metal-isotopes, represent the individual cells from the tissue sample. These two images suggest that the white spots correspond to cells (cell nuclei) and locations of a darker gray or black correspond to tissue background.

The point-wise maximum and resulting image plot produced an image of white spots that represent cells on the slide with a black background. It can been seen here that the black background, ie. minimum pixel intensity, does not occupy much of this image, indicating that in locations which display a lighter gray, still represent pixels where DNA markers were targeted and measured. The finalized cell segmentation in the fourth image uses Otsu's method to create a binary mask of black and white where all pixel intensities higher than the calculated threshold value are assumed to be cell nuclei. Cells contained within the mask are represented as a true white color whereas all other pixels with an intensity value lower than the calculated threshold are assumed to be background containing little to no DNA and colored black. Comparing both the point-wise max and finalized cell nuclei mask, it can be seen that in areas that were previously a lighter gray in the third image have been assigned as background in the fourth. This suggests that Otsu's method may have over-segmented the image, assigning pixels of lower intensity values, but still with authentic DNA marker measurements to the background.

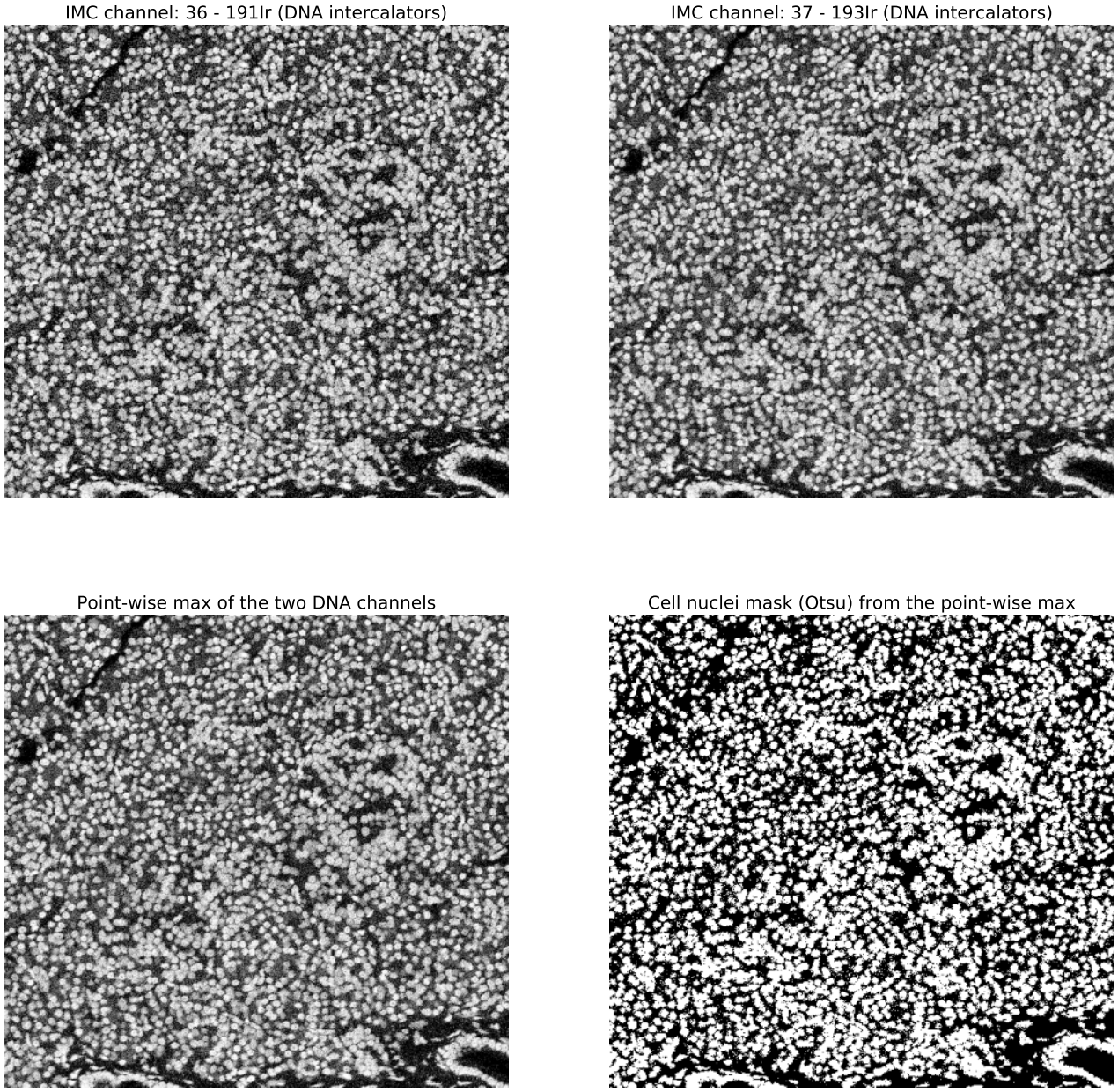

Figure 4: Generated images of IMC channels 36 and 37 are displayed as the top two images, using a gray-scale colormap to represent measured pixel intensities across all pixel locations. The calculated point-wise maximum of the two DNA channels is displayed in the bottom left and cell nuclei segmentation using the histogram-based Otsu's method in the bottom right. Pixel locations with intensity values higher than the calculated threshold are displayed as white assigned to the foreground, where pixel locations with intensity values lower than the calculated threshold are displayed in black and assigned as background (figure produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-3.ipynb`).

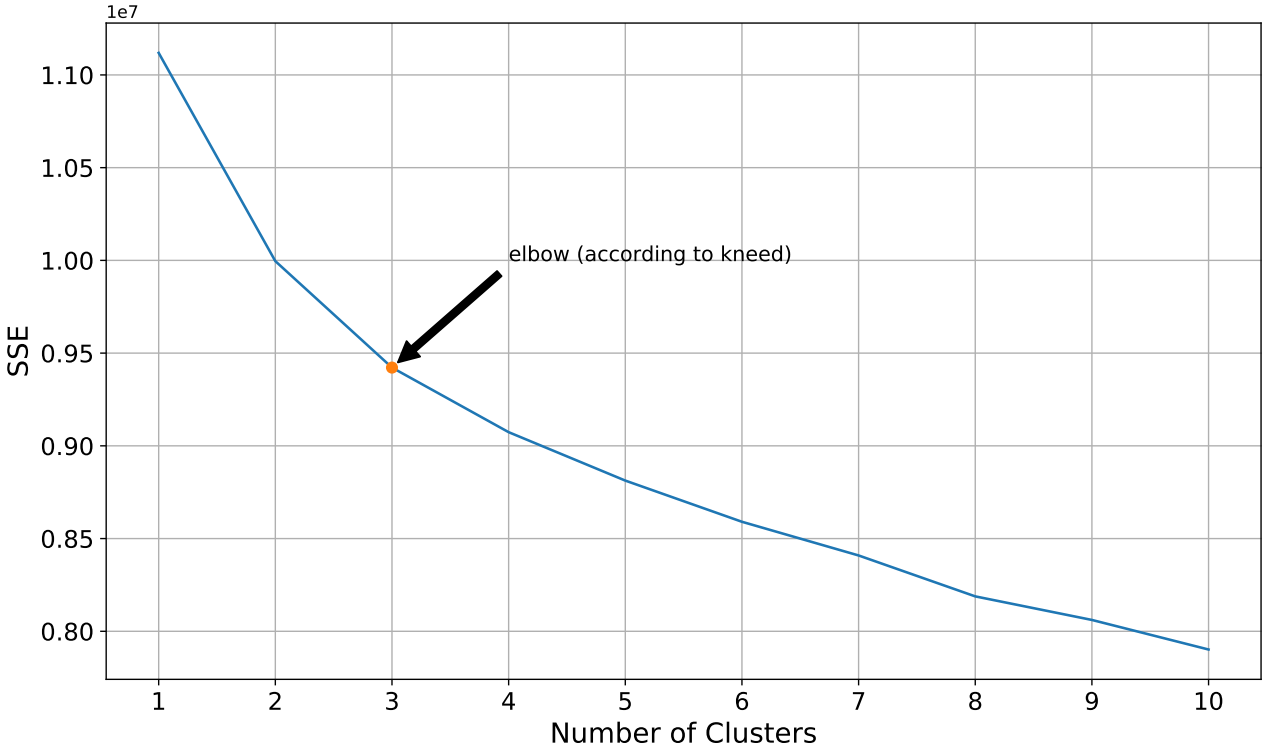

Figure 5: Choosing the appropriate number of clusters using the *elbow* (sweet spot) method as implemented in `kneed`. The elbow curve is shown as the calculated sum of squares error (SSE) on the y-axis and the corresponding number of clusters on the x-axis. The inflection point which indicates the optimal number of clusters is the maximum point of curvature, marked by an arrow (figure produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-2.ipynb`).

Choosing the appropriate number of clusters using the `elbow` and the `silhouette` methods, we found that the elbow method produced a curve with a difficult to distinguish inflection point. The curve as shown in Fig. 5, does illustrate a drop in SSE as the number of clusters increases from 1 to 3, but is not so clear as to be able to make an immediate visual distinction of a suitable number of clusters. Using the Python package `kneed`, the optimum point of curvature was found to correspond to 3 clusters. The time used to compute the SSE values for each cluster was fast with a wall time of 3 minutes.

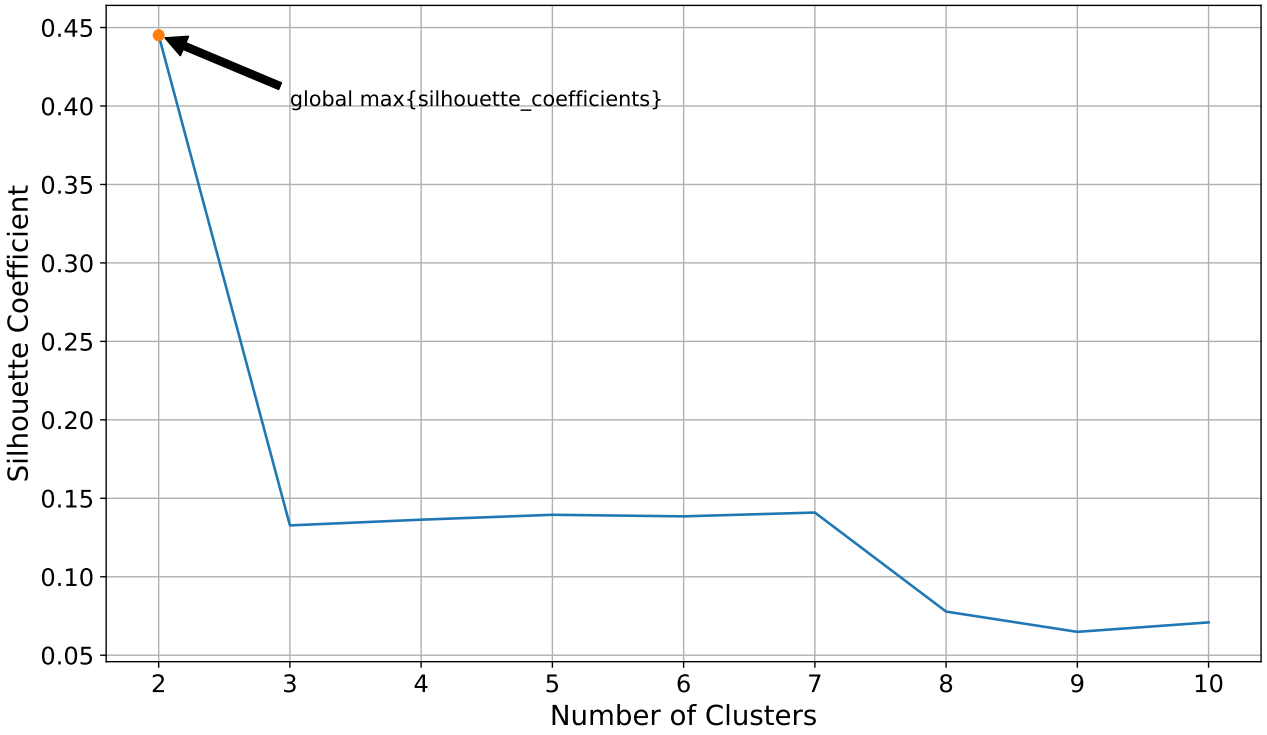

Figure 6: Choosing the appropriate number of clusters using the *silhouette* method as implemented in `scikit-learn`. Silhouette scores represent the y-axis and corresponding number of clusters on the x-axis. Scores nearest to 1 equate to better cluster separation in the dataset, where the maximum silhouette coefficient here is indicated by an arrow. Compute time (multi-core CPU) was $\approx$ 2 hr (figure produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-2.ipynb`).

As shown in Fig. 6, $K = 2$ produces the highest silhouette score, indicating that 2 clusters produce the best intra-cluster similarity and inter-cluster separation. Values of K higher than 2 produced scores much closer to zero, which can indicate overlapping clusters and insufficient separation. The highest score however, still remains below half of an ideal silhouette score, indicating unsatisfactory clustering across the dataset. The wall time used to compute the different silhouette scores was much more time consuming, taking approximately 2 hours. Because of the differing results between the methods of evaluation, it was determined that a max silhouette score less than 0.5 was not ideal and instead $K = 3$ was selected as per the elbow method, despite the opposing conclusion reached by the silhouette method.

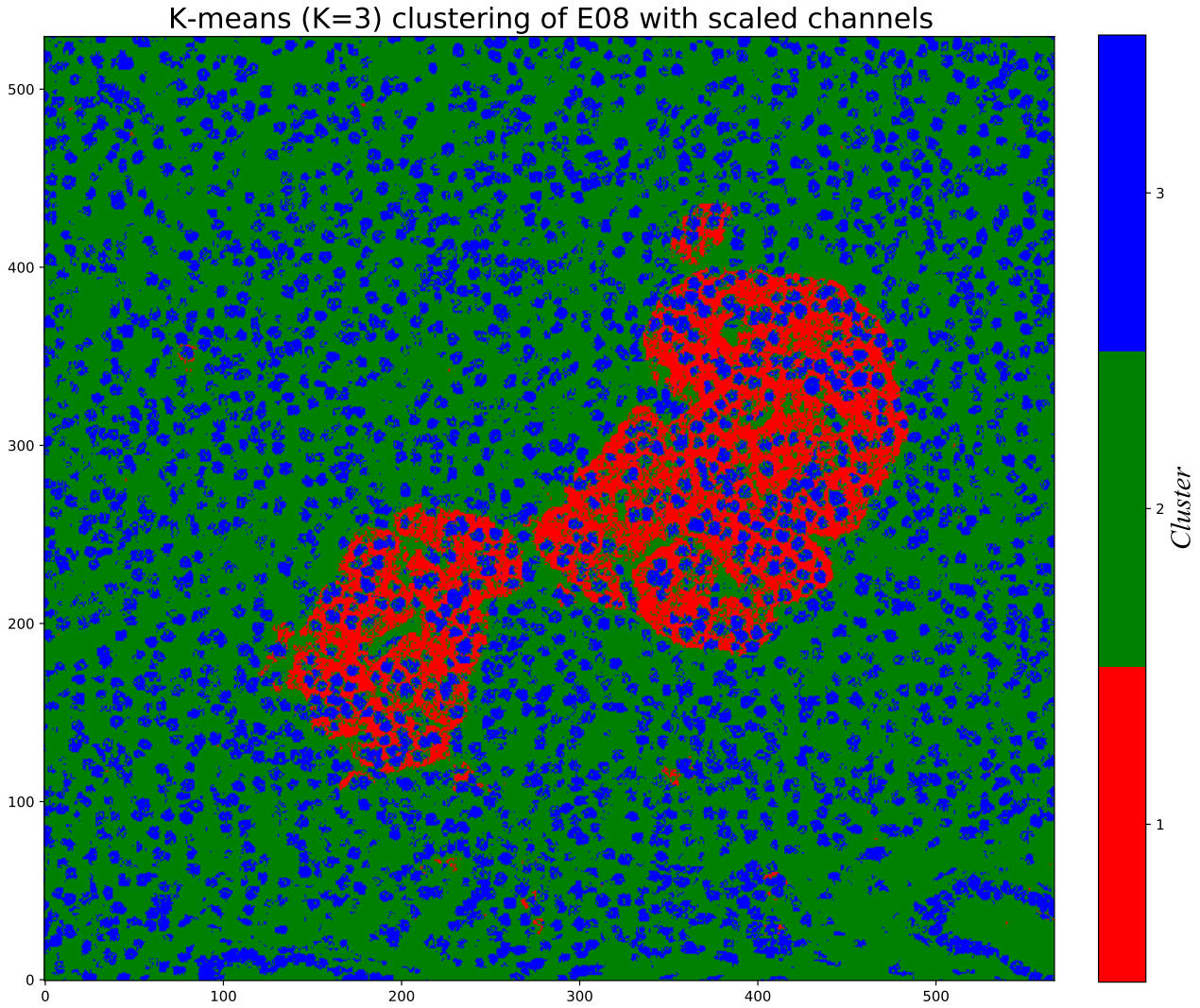

Figure 7: K-means clustering using the *elbow* (sweet spot) heuristic value of $K = 3$ where each cluster is assigned a different color in the generated image. Cluster 1 illustrates CyTOF features associated with a pancreatic islet, cluster 2 is characterized as background, and cluster 3 represents pixel features of cell nuclei (figure produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-2.ipynb`).

Clustering results in Fig. 7 show K-means segmentation for $K = 3$ where the three distinct clusters are colored red, green, and blue, each depicting a set of pixel features obtained from the CyTOF metal-isotope measurements. The size of the segmented image is 530×567 pixels with a total number of clustered samples equal to 300510. Because IMC employs non-optical imaging techniques, morphological features of the tissue are visually limited to rough outlines. In this image, cluster 2 resembles a background, and could be understood to be a grouping of features with lower pixel intensity as a direct result of the small number of clusters assigned to the algorithm. Cluster 3 is displayed in a similar manner to that of segmentation by thresholding, depicting areas of the tissue targeted by antibodies directed towards DNA biomarkers, where the blue spots can be interpreted to represent cell nuclei. Cluster 1 however forms a distinct shape of a pancreas islet in the middle of the image, indicating that cluster 1 consists of a grouping of CyTOF-pixel signatures aimed at

biomarkers abundantly found in islet cells.

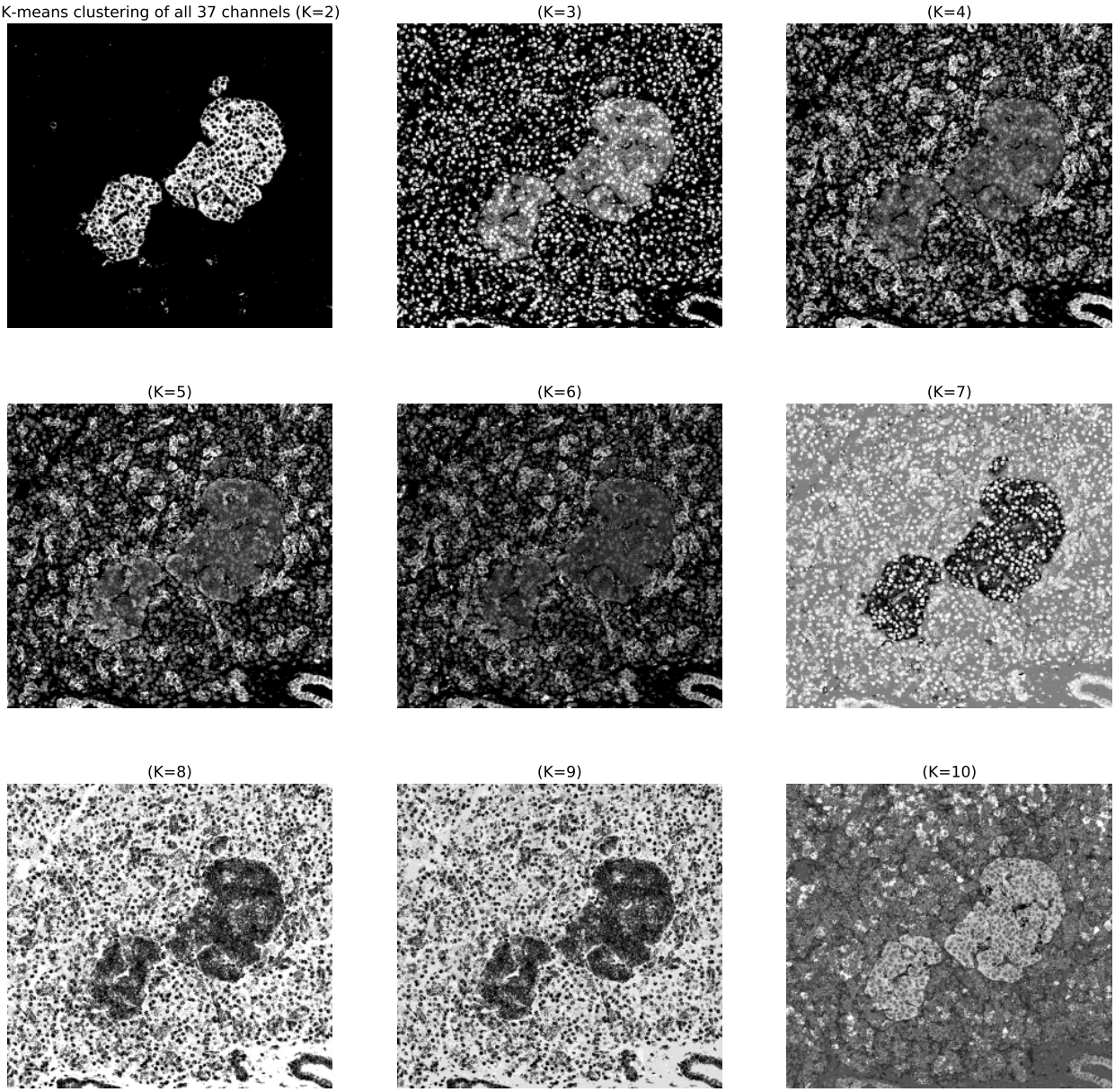

Figure 8: K-means clustering with varying $K = 2, \ldots, 10$ where each cluster is given a gray level. It is illustrated here that a differing number of set clusters in the K-means algorithm produces distinctly different results using the same dataset (figure produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-2.ipynb`).

Fig. 8 shows the additional runs of K-means clustering and how the algorithm produces varying results with a differing number of set clusters. $K = 2$ shows a distinct image of a pancreas islet, displaying spots within the islet that represent cell nuclei. While this clustering result produces an anatomically relevant image, individual cells and cell nuclei outside of the islet are not visible (assigned to "background"). $K = 3$ shows the same segmentation results as in Fig. 7 with a different colormap and does not offer any new interpretations. Although, the assigned colors may in some cases be more visually comfortable to those who prefer a black background. As the number of clusters increases beyond $K = 3$, the results become much more varied. As shown in results from

$K = 4$ to $K = 6$, individual cells become difficult to distinguish as well as that islet features become much more obscure. Despite this, these results bring a small previously unseen feature into the foreground. The bottom right hand corner of each of these three generated images show a collection of cell nuclei in a oval shape similar to a pancreatic duct. Surrounding the suspected nuclei is another clustered set of features, assigned a different color which suggests that this may represent CyTOF features or targeted proteins found in cytoplasm. Clustering results from $K = 7$ to $K = 9$ bring back the islet features into the foreground but again, individual cells and cell nuclei both within and outside of the islet are not as clearly defined where more features are being clustered in the adjacent areas. Where $K = 10$ all distinct cell nuclei features are absent outside of the islet, and can be thought to have been clustered into various groups losing the appearance of segmentation.

5.4. Correlation matrix of CyTOF features

The correlation matrix in Fig. 9 is shown as the lower triangle of the originally generated symmetric plot. On the right-hand side, the color gradient indicates the calculated correlation values between the different CyTOF features where lighter colors represent negative correlation and darker colors represent positive values. The numerical value for each isotope pair is also illustrated in each box rounded to 2 decimal places. Several areas on the correlation plot show more significant levels of association than others. The two DNA targeting IMC channels (191Ir and 193Ir) show a very strong positive correlation, markers 145Nd (CD99) and 160Gd (SYP), both targets of endocrine cells exhibit a strong positive correlation, while 144Nd (PCSK2) and 145Nd show a moderate positive correlation. Still, the bulk of the values fall under negligible or a weak positive correlation. The values displayed here can help us predict how the CyTOF features change in relation to each other, where a positive correlation represents that as measurements of one isotope increase so does the other. Indicating that increased protein expression of one variable in a specific area of tissue corresponds with an increase in protein expression of the other.

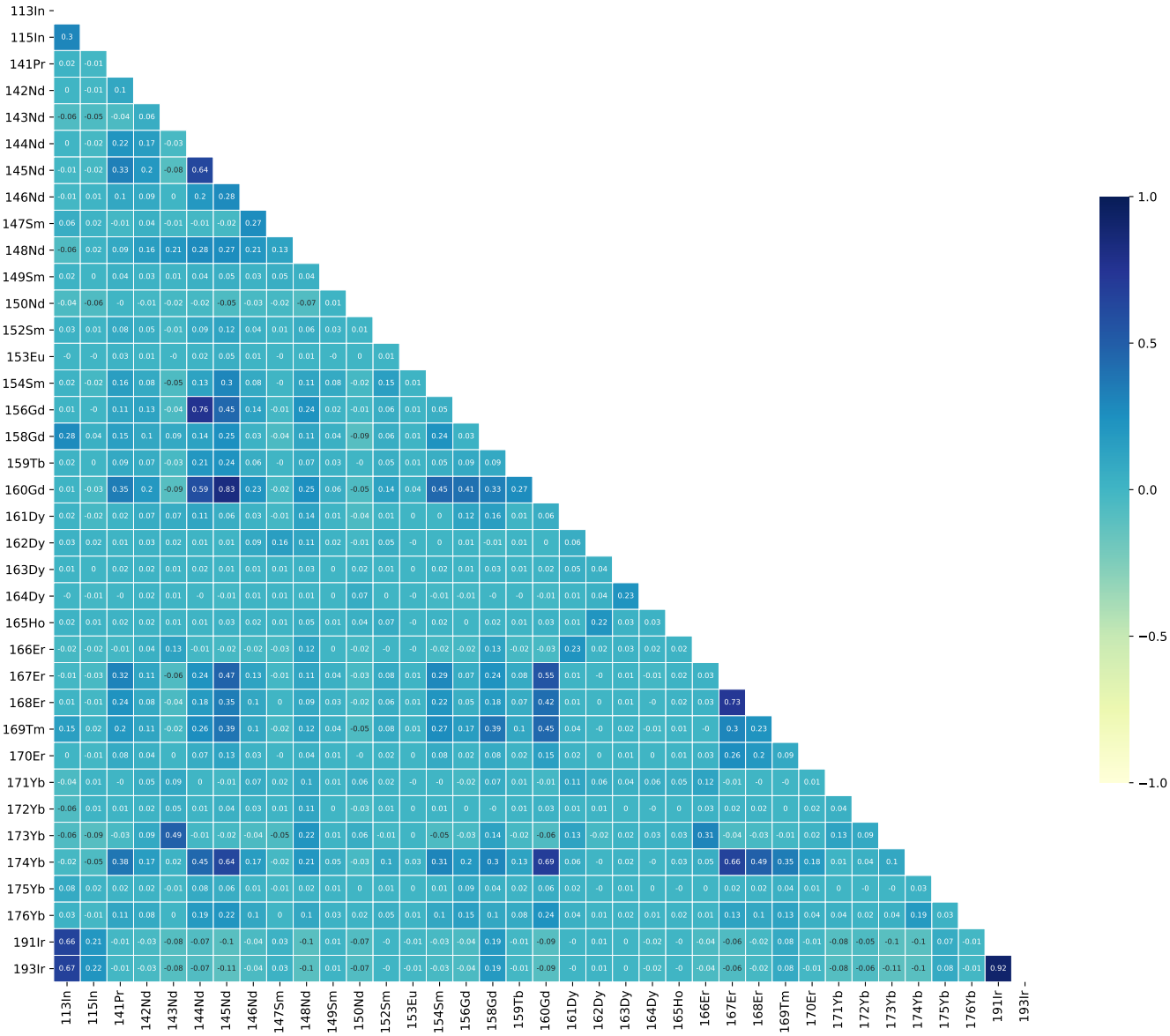

Figure 9: Annotated heatmap of Pearson correlation matrix, calculated between every pair in the collection of 37 channel images (figure produced in the Jupyter notebook `read-analyze-case-6126-E08-ajia-4.ipynb`).

6. Discussion

Both simple and efficient, histogram thresholding such as Otsu's binarization show the capacity to compute and generate cell-nuclei masks, which can in turn allow for further single-cell analysis with spatial information. However, the presence of noise or artifacts may result in segmentation error where thresholding algorithms only consider pixel intensity and not the relationship between pixels or objects in the image [41]. It should also be assumed that cell populations with dim biomarker expression (eg. low pixel intensity of DNA channel signals) may not be as efficiently observed and thereby classified as background using thresholding techniques [42]. Despite these aforementioned limitations, Otsu's method specifically is among the more commonly used and well-established techniques, demonstrating a computationally fast approach to IMC single-cell segmentation.

Additionally, unsupervised clustering by K-means has indicated successful and tissue relevant cell segmentation with the added advantage that it is suitable for large datasets and flexible when choosing a set number of clusters [43]. Although, setting the initial number of clusters must be done by either having prior knowledge of existing clusters within the dataset or through evaluation methods. This crucial step has shown the means to considerably alter image segmentation outcome and the resulting interpretations. K-means clustering seems though to differ from thresholding algorithms in one major way; it can produce images with more anatomically relevant detail. As seen in both K-means clustering figures, tissue morphology becomes clearer and islet features become more prominent. This advantage alone is pertinent in the analysis of specific tissue microenvironments where researchers are interested in more than segmented cell outlines, but the relationships between protein expression and spatial organization in tissue.

Moreover, the generated correlation heatmap illustrates Python's ability to analyze high dimensional IMC data, revealing potential biological relationships between CyTOF features. It is though important to stress that observed correlation does not always indicate causation and should be interpreted in conjunction with other statistical models and clinically relevant data [40].

Despite a seemingly successful segmentation process, further comparison using more refined, specifically designed software must be made if accuracy of the methods is to be determined. Similarly, mainstream supervised programs are equipped to classify substantially more pixel features and contain already a number of tested and published algorithms which help maintain reliability and accuracy of downstream analysis [19, 21].

7. Conclusion

To properly understand the roll complex tissue function plays it becomes necessary to analyze localized phenotypic characteristics at sub-cellular resolution, but in turn requires specialized software in order to glean insight into the data. IMC bridges this gap and utilizes machine learning algorithms to derive useful information on pattern recognition, feature analysis, and data classification. Although biomedical imaging techniques such as IMC typically use carefully tailored supervised software to perform single-cell segmentation [44], a much more basic approach using thresholding or unsupervised clustering algorithms can be used. The Python libraries utilized here have also shown the capacity to manage and interpret the high dimensional IMC data, manipulating it into easily readable formats and visualizations which help perform initial analysis. Further experimentation and optimization of these techniques would be preferable where perhaps these from scratch methods can offer a more personalized and adaptive approach to IMC data examination.

Acknowledgements

I would like to thank Professor Arvid Lundervold (University of Bergen) for his contributions in the experimental part of this project, his expertise in the area of computational imaging, and knowledge of machine learning as well as for the useful discussions. This work has been conducted in collaboration with the Computational Imaging and Machine Learning group at the Mohn Medical Imaging and Visualization Centre (`https://mmiv.no/machinelearning`) and with input from users of the IMC technology at University of Bergen.

References

[1] M. H. Spitzer, G. P. Nolan, Mass cytometry: Single cells, many features, Cell 165 (2016) 780–791.

[2] M. Roederer, Spectral compensation for flow cytometry: Visualization artifacts, limitations, and caveats, Cytometry 45 (2001) 194–205.

[3] Q. Chang, O. I. Ornatsky, I. Siddiqui, A. Loboda, V. I. Baranov, D. W. Hedley, Imaging mass cytometry., Cytometry. Part A : the journal of the International Society for Analytical Cytology 91 (2017) 160–169.

[4] G. Han, M. H. Spitzer, S. C. Bendall, W. J. Fantl, G. P. Nolan, Metal-isotope-tagged monoclonal antibodies for high-dimensional mass cytometry, Nature protocols 13 (2018) 2121–2148.

[5] H. Baharlou, N. P. Canete, A. L. Cunningham, A. N. Harman, E. Patrick, Mass cytometry imaging for the study of human diseases-applications and data analysis strategies., Frontiers in immunology 10 (2019) 2657.

[6] K. R. Atkuri, J. C. Stevens, H. Neubert, Mass cytometry: A highly multiplexed single-cell technology for advancing drug development, Drug Metabolism and Disposition 43 (2015) 227–233.

[7] M. N. Gurcan, L. E. Boucheron, A. Can, A. Madabhushi, N. M. Rajpoot, B. Yener, Histopathological image analysis: a review, IEEE reviews in biomedical engineering 2 (2009) 147–171.

[8] K. M. McKinnon, Flow cytometry: An overview, Current protocols in immunology 120 (2018) 5.1.1–5.1.11.

[9] R. L. McCarthy, A. D. Duncan, M. C. Barton, Sample preparation for mass cytometry analysis, Journal of visualized experiments : JoVE (2017) 54394.

[10] A. Somarakis, V. Van Unen, F. Koning, B. P. F. Lelieveldt, T. Hollt, Imacyte: Visual exploration of cellular microenvironments for imaging mass cytometry data., IEEE transactions on visualization and computer graphics (2019).

[11] R. Straus, A. Carew, D. Sandkuijl, T. Closson, V. Baranov, A. Loboda, Analytical figures of merit for a novel tissue imaging system, Journal of Analytical Atomic Spectrometry 32 (2017) 1044–1051.

[12] E. Gerdtsson, M. Pore, J. A. Thiele, A. S. Gerdtsson, P. D. Malihi, R. Nevarez, A. Kolatkar, C. R. Velasco, S. Wix, M. Singh, A. Carlsson, A. J. Zurita, C. Logothetis, A. A. Merchant, J. Hicks, P. Kuhn, Multiplex protein detection on circulating tumor cells from liquid biopsies using imaging mass cytometry, Converg Sci Phys Oncol 4 (2018).

[13] N. Damond, S. Engler, V. R. T. Zanotelli, D. Schapiro, C. H. Wasserfall, I. Kusmartseva, H. S. Nick, F. Thorel, P. L. Herrera, M. A. Atkinson, B. Bodenmiller, A map of human type 1 diabetes progression by imaging mass cytometry., Cell metabolism 29 (2019) 755–768.e5.

[14] A. Torkamani, K. G. Andersen, S. R. Steinhubl, E. J. Topol, High-definition medicine, Cell 170 (2017) 828–843.

[15] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.

[16] G. Barbastathis, A. Ozcan, G. Situ, On the use of deep learning for computational imaging, Optica 6 (2019) 921–943.

[17] B. J. Erickson, P. Korfiatis, Z. Akkus, T. L. Kline, Machine learning for medical imaging, Radiographics : a review publication of the Radiological Society of North America, Inc 37 (2017) 505–515.

[18] V. Zanotelli, ndamond, M. Strotton, BodenmillerGroup/ImcSegmentationPipeline: IMC Segmentation Pipeline, 2020.

[19] A. E. Carpenter, T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, P. Golland, D. M. Sabatini, Cellprofiler: image analysis software for identifying and quantifying cell phenotypes, Genome Biology 7 (2006) R100.

[20] T. Tsujikawa, G. Thibault, V. Azimi, S. Sivagnanam, G. Banik, C. Means, R. Kawashima, D. R. Clayburgh, J. W. Gray, L. M. Coussens, Y. H. Chang, Robust cell detection and segmentation for image cytometry reveal th17 cell heterogeneity, Cytometry Part A 95 (2019) 389–398.

[21] S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, K. Eren, J. I. Cervantes, B. Xu, F. Beuttenmueller, A. Wolny, C. Zhang, U. Koethe, F. A. Hamprecht, A. Kreshuk, ilastik: interactive machine learning for (bio)image analysis, Nature Methods 16 (2019) 1226–1232.

[22] N. Dhanachandra, K. Manglem, Y. J. Chanu, Image segmentation using k -means clustering algorithm and subtractive clustering algorithm, Procedia Computer Science 54 (2015) 764–771.

[23] M. Chinki, C. Mrs, S. Chaturvedi, A. Khurshid, An approach to image segmentation using k-means clustering algorithm, International Journal of Artificial Intelligence and Neural Networks 2 (2012) 197 – 201.

[24] M. Zhao, J. An, H. Li, J. Zhang, S.-T. Li, X.-M. Li, M.-Q. Dong, H. Mao, L. Tao, Segmentation and classification of two-channel c. elegans nucleus-labeled fluorescence images, BMC Bioinformatics 18 (2017) 412.

[25] P. Yan, X. Zhou, M. Shah, S. T. C. Wong, Automatic segmentation of high-throughput rnai fluorescent cellular images, IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society 12 (2008) 109–117.

[26] T. Y. Goh, S. N. Basah, H. Yazid, M. J. Aziz Safar, F. S. Ahmad Saad, Performance analysis of image thresholding: Otsu technique, Measurement 114 (2018) 298–307.

[27] E. Meijering, Cell segmentation: 50 years down the road [life sciences], IEEE Signal Processing Magazine 29 (2012) 140–145.

[28] D. Schapiro, H. W. Jackson, S. Raghuraman, J. R. Fischer, V. R. T. Zanotelli, D. Schulz, C. Giesen, R. Catena, Z. Varga, B. Bodenmiller, histocat: analysis of cell phenotypes and interactions in multiplex image cytometry data, Nature Methods 14 (2017) 873–876.

[29] A. S. Lundervold, A. Lundervold, An overview of deep learning in medical imaging focusing on mri, Zeitschrift für Medizinische Physik 29 (2019) 102–127.

[30] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, arXiv: Quantum Physics (2013).

[31] C. McQuin, A. Goodman, V. Chernyshev, L. Kamentsky, B. A. Cimini, K. W. Karhohs, M. Doan, L. Ding, S. M. Rafelski, D. Thirstrup, W. Wiegraebe, S. Singh, T. Becker, J. C. Caicedo, A. E. Carpenter, Cellprofiler 3.0: Next-generation image processing for biology, PLOS Biology 16 (2018) e2005970.

[32] K. Smith, F. Piccinini, T. Balassa, K. Koos, T. Danka, H. Azizpour, P. Horvath, Phenotypic image analysis software tools for exploring and understanding big image data from cell-based assays, Cell Systems 6 (2018) 636–653.

[33] A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: A review, ACM Comput. Surv. 31 (1999) 264–323.

[34] J. M. Perkel, Why jupyter is data scientists' computational notebook of choice, Nature 563 (2018) 145–146.

[35] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, J. development team, Jupyter notebooks - a publishing format for reproducible computational workflows, in: F. Loizides, B. Scmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas, IOS Press, 2016, pp. 87–90.

[36] J. VanderPlas, Python data science handbook : essential tools for working with data, O'Reilly, Sebastopol, Calif., 2017.

[37] S. L. Bangare, A. Dubal, P. S. Bangare, S. Patil, Reviewing otsu's method for image thresholding, International Journal of Applied Engineering Research 10 (2015) 21777–21783.

[38] C. Yuan, H. Yang, Research on k-value selection method of k-means clustering algorithm, J 2 (2019) 226–235.

[39] K. Arvai, Kneed, `https://github.com/arvkevi/kneed`, 2020.

[40] P. Schober, C. Boer, L. A. Schwarte, Correlation coefficients: Appropriate use and interpretation, Anesth Analg 126 (2018) 1763–1768.

[41] A. K. Chaubey, Comparison of the local and global thresholding methods in image segmentation, World Journal of Research and Review 2 (2016).

[42] M. E. Ijsselsteijn, R. van der Breggen, A. Farina Sarasqueta, F. Koning, N. F. C. C. de Miranda, A 40-marker panel for high dimensional characterization of cancer immune microenvironments by imaging mass cytometry, Frontiers in Immunology 10 (2019).

[43] A. Kumar, A. Tiwari, A comparative study of otsu thresholding and k-means algorithm of image segmentation, International Journal of Engineering and Technical Research 9 (2019).

[44] L. Shamir, J. D. Delaney, N. Orlov, D. M. Eckley, I. G. Goldberg, Pattern recognition software and techniques for biological image analysis, PLOS Computational Biology 6 (2010) e1000974.