Tutorial for multiplexed image analysis using TRACERx-PHLEX

Alastair Magness, Emma Colliver, Katey S. S. Enfield, Mihaela Angelova
Cancer Evolution and Genome Instability Laboratory, The Francis Crick Institute, London, UK.

Table of contents

TRACERx-PHLEX performs deep learning-based cell segmentation (deep-imcyto), automated detection of spatially resolved protein expression and cell-type annotation (TYPEx), and interpretable spatial analysis (Spatial-PHLEX), as three independent but interoperable modules. The pipeline outputs spatially resolved single-cell identities, cell subtype densities within tissue compartments, marker positivity calls, and spatial metrics such as cellular barrier scores, along with summary graphs and spatial visualisations. PHLEX was developed using imaging mass cytometry (IMC) in the TRACERx study, validated using published CODEX, IMC and orthogonal data and benchmarked against state-of-the-art approaches. It was evaluated on different tissue types, tissue fixation conditions, image sizes and antibody panels. As PHLEX is an automated and containerised Nextflow pipeline, manual assessment, programming skills or pathology expertise are not essential. PHLEX offers an end-to-end solution for the biological imaging community in a growing field of highly multiplexed data and helps provide clinically relevant insights.

The TRACERx-PHLEX pipeline is available on GitHub along with instructions on how to run it on the PHLEX test dataset:

https://github.com/FrancisCrickInstitute/TRACERx-PHLEX

The deep-imcyto core nuclear prediction model is available for use outside of the deep-imcyto Nextflow pipeline:

https://github.com/FrancisCrickInstitute/py-imcyto

All software dependencies were integrated in the Nextflow pipeline and deposited as containers on the public repository dockerhub.

The TRACERx Nuclear IMC segmentation dataset, deep-imcyto's trained neural network model weights, and the PHLEX test dataset can be downloaded from Zenodo:
https://zenodo.org/record/7973724

The cell-by-marker intensity table for the publicly available CODEX dataset on colorectal cancer was provided in the Extended Data of the original study by Schuerch *et al.*[1]. CODEX data on intestine (HuBMAP) and Barrett's esophagus (BE) were downloaded from the online repository Dryad indicated in the original publication [2].

A detailed user guide and documentation on PHLEX functionalities is available on https://tracerx-phlex.readthedocs.io/

Applications of TRACERx-PHLEX

TRACERx-PHLEX was developed for end-to-end analysis of IMC data and applied to the TRACERx non-small cell lung cancer study. Its modular design and the generic input format of its modules TYPEx and Spatial-PHLEX make it applicable to other highly multiplexed imaging data types, including PhenoCycler/co-detection by indexing (CODEX) [3].

We developed and evaluated PHLEX using IMC on 236 formalin-fixed paraffin-embedded (FFPE) tissue microarray (TMA) samples from 83 patients with early-stage treatment-naïve non-small cell lung cancer (NSCLC) from the TRACERx 100 cohort (Enfield *et al.*, under review) [4]. We used two 35-plex antibody panels: the *T cells & Stroma* panel for assessing the differentiation states of T cells [5] and capturing non-immune stromal cells, and the Pan-Immune panel for interrogating innate and adaptive immunity cell subtypes. The panels targeted immune, stromal and epithelial cells along with a range of checkpoint molecules, phenotypic, functional and metabolic markers (**Supplementary Table S1**). We segmented and typed more than 3.16M cells for each panel. PHLEX allowed analyses of immune evasion mechanisms within a single workflow and identified and quantified T cell differentiation and exhaustion states in tumour tissue (Enfield *et al.*, under review). Using PHLEX, we identified potential physical αSMA^{+} fibroblast barriers to tumour-T cell engagement and their association with cold tumour microenvironments. Finally, PHLEX allowed systematic characterisation of the spatial heterogeneity of cell subtypes, multicellular spatial structures and broad TME categories [6].

Beyond the TRACERx 100 IMC cohort, we also demonstrate the application of individual PHLEX modules to other datasets. We evaluated the utility of deep-imcyto segmentation on different cancer types and experimental settings through application to two datasets of IMC images of breast cancer [7,8], as well as fresh-frozen mouse lung tissue [9]. When applied as a standalone module, TYPEx can be used for IMC as well as other multiplexed imaging technologies. We evaluated and benchmarked the performance of TYPEx on FFPE-embedded TMAs and fresh frozen tissue sections obtained with CODEX. We demonstrated its performance on publicly available CODEX datasets [1,2] and analysed cell phenotypes on samples from colorectal cancer, healthy intestine and Barrett's esophagus.

Expertise needed

TRACERx-PHLEX is straightforward to set up and run. It requires basic knowledge of command line use to navigate to and edit the bash script for initiating Nextflow. Programming skills and pathology expertise are not essential. To get the most out of PHLEX a user requires the following:

Configuration of TYPEx requires basic understanding of markers expressed on different cell subtypes and lineages and any potential non-specific binding of antibodies in the panel representing major cell-lineage and subtype-specific markers.

To develop a Multiplexed Consensus Cell Segmentation (MCCS) workflow to be used for deep-imcyto in CellProfiler mode, a basic level of knowledge of cell imaging is required to allow a user to select cytoplasmic and membrane segmentation markers for represented major cell lineages and to discriminate between imaging artefacts and genuine cell and tissue features. The CellProfiler software used to design and MCCS workflow is straightforward for a new user to implement, but prior experience could still be helpful. No programming expertise is required to implement an MCCS procedure. Fuller details of considerations and caveats for designing and implementing an MCCS procedure are provided in the **Tutorial 4**.

Limitations

At present the Nextflow implementation of deep-imcyto requires a system with CUDA-enabled GPU hardware to run. However for users unable to meet this requirement, or those who wish to use the underlying Python model, there is py-imcyto, a streamlined Python repository that utilises the same trained model as deep-imcyto.

To detect marker positivity, TYPEx assumes that a marker is expressed on a subset of cells, which may not be fulfilled, for example, for β2-microglobulin, which is expressed on all nucleated cells with the exception of some tumour cells.

By default, TYPEx relies on T cell markers to call marker positivity. In case of sparse presence of T cells in the analysed cohort, for example, including only cold TMEs, finding the optimal D-score cutoff may be affected. Such cases can be recognised when the curve is not concave upward for both the low- or high-confidence groups. As a solution, TYPEx allows any markers and combinations thereof defining rare and high-frequency cell populations to be specified in the *typing_params.json* config file.

The limited reproducibility of clustering the entire cohort of cells is mitigated by TYPEx at the level of major cell lineage, by clustering within the same major cell lineage and confidence group. However, there may still be a degree of cell mixing within a given cell lineage at the level of marker positivity/cell subpopulations.

Due to their computational expense, Spatial-PHLEX barrier scoring algorithms are GPU-accelerated and, thus, require the use of GPU hardware. However, Spatial-PHLEX may still be run in spatial clustering mode alone.

Configuration options and workflows in TRACERx-PHLEX

Workflow options for segmentation in deep-imcyto

deep-imcyto has three modes of operation: *simple*, CellProfiler, and QC, which the user must set to produce the anticipated results.

- Workflow 1: QC: quality control of raw IMC data, with optional processing steps.
- Workflow 2: *simple* segmentation: segmentation of nuclei with the deep-imcyto nucleus segmentation model, followed by pixel expansion to approximate cellular boundaries.
- Workflow 3: Multiplexed Consensus Cell Segmentation: segmentation of nuclei with the deep-imcyto nucleus segmentation model, followed by execution of a custom CellProfiler pipeline designed to take nuclear predictions as input.

Image QC

deep-imcyto's specialised QC mode, facilitates swift access to individual channels within IMC data to ensure quality control or review. It achieves this by dividing input .mcd, .txt, and .tiff files into constituent channel images from imaged ROIs in a fast and parallelised fashion. Upon choosing specific preprocessing options, such as spillover correction, hot pixel removal, or the application of custom preprocessing steps specified in a CellProfiler .cppipe file, the chosen preprocessing steps will be executed and presented as output for manual review during the QC run.

We used deep-imcyto QC during antibody panel validation and optimisation for application in the TRACERx 100 IMC study, inspecting individual channels to ensure sufficient quality of signal across all our markers.

simple segmentation

simple segmentation mode produces an approximate whole cell segmentation by dilating precise predicted nuclei with a user-defined pixel count. This approach generates a segmentation mask for every cell in the images and conducts measurements of marker intensity and morphometry. Additionally, it generates informative single-cell spatial plots showing mean, standard deviation, and median marker intensities, providing users with rapid insights into the spatial distribution of specific protein markers. A table containing the nearest spatial neighbours of each cell object is also generated for subsequent nearest-neighbour analysis.

Cellprofiler mode

The CellProfiler segmentation mode produces a whole-cell segmentation by harnessing intensity information in cytoplasmic and membrane channels to guide propagation onto cell boundaries from deep-learning nuclei. Unlike in the fully automated *simple* segmentation mode, in CellProfiler mode, a user must provide a pre-designed procedure to deep-imcyto as an executable CellProfiler pipeline. The MCCS procedure, for example, takes advantage of the experimenter's knowledge about cell lineages and marker expression to enhance whole cell segmentation across diverse cell types within complex tissues, but requires the user to develop the CellProfiler procedure themself. Further guidance on developing an MCCS workflow for an IMC panel is provided in **Tutorial 4: Designing an MCCS workflow in CellProfiler**.

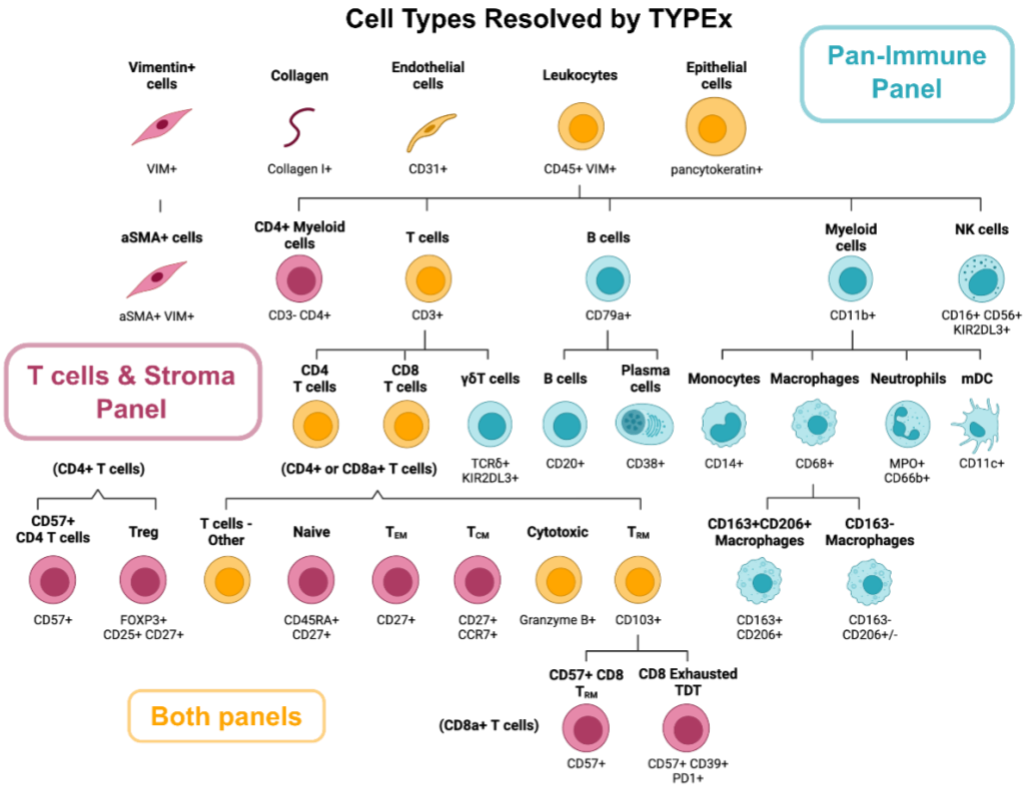
Customising the cell-type definitions to the antibody panel for cell phenotyping

TYPEx can be customised with input parameters and configuration files, allowing it to automate the cell phenotyping analysis without the need for manual adjustments of clusters or post hoc reannotation. A required input for TYPEx is a cell-by-marker intensity matrix, which can be provided by specifying the path to the deep-imcyto release folder, or, in standalone mode, the path to a tab-separated file. A file with definitions of cell lineages and subtypes tailored to the user’s antibody panel is also required.

The cell-type definitions file is the critical input file for cell phenotyping analyses with TYPEx. Several template files are provided with the TYPEx distribution designed for the antibody panels analysed in Magness *et al.*, which can be adjusted to match a new antibody panel. The cell type definitions file includes a list of major cell lineages and a list of the cell subtypes targeted by the

panel of interest. The key step is selection of the markers that define these cell lineages and subtypes, as shown, for example, for the T cells & Stroma and the Pan-Immune panels (**Figure 1a**). To identify the major cell lineages and the associated lineage-specific markers, the first step is to select the markers with significantly higher intensities expected in one cell lineage, which in the case of the T cells & Stroma panel included CD4 and CD8 T cells (CD3, CD4, CD8), aSMA+ cells (aSMA), Endothelial cells (CD31), and Epithelial cells (panCK). The next step is to define the cell lineages targeted by markers in the panel that can also be expressed in other cell types, such as Leukocytes - Other (CD45), T cells - Other (CD3) and Vimentin+ cells (Vimentin). Such non-specific markers can also be added to the definitions of other cell lineages that express them only when they are hierarchically related. Therefore, CD45 and CD3 were added to the definitions of CD4 and CD8 T cells, resulting in the lineage hierarchy illustrated in **Figure 1a-b**. For this reason, although Vimentin can also be expressed by immune cells, it was included only in the definitions of stromal cells.

a



b

```
"major_markers":{
  "Leukocytes":{
    "Leukocytes - Other":["CD45"],
    "T cells":{
      "T cells - Other":["CD45", "CD3"],
      "CD8 T cells":["CD45", "CD3", "CD8a"],
      "CD4 T cells":["CD45", "CD3", "CD4"]
    }
  },
  "Stroma":{
    "Vim+ cells":["vimentin"],
    "Endothelial cells":["vimentin", "CD31"],
    "aSMA+ cells":["vimentin", "aSMA"]
  },
  "Epithelial cells":["panCK"]
},
```

Figure 1: Antibody panels and cell subtype definitions in the TRACERx IMC study.

a, Data generated from all antibody stains was used in the TYPEx phenotyping module. Cell types resolved using TYPEx are represented hierarchically with defining protein markers indicated. Cell types are coloured according to their associated antibody panel (T cells and Stroma, red; Pan-Immune, blue), and those cells identified by both antibody panels are coloured in gold. **b**, Example major cell lineage definitions provided to TYPEx in the input config file specified with *–annotation_config* for T cells & Stroma panel. The repetition of markers such as CD45 and CD3 is considered to infer the cell hierarchy for CellAssign and for the cell subtype annotation.

The main considerations that differ between cell lineage and subtype definitions is the specificity of the markers. For the lineage definitions, the aim is to identify the most likely cell lineage, and the specificity of the markers is therefore not prioritised. For example, in the Pan-Immune panel, both CD4 T cells and Macrophages are defined as major cell lineages. Although CD4 can be expressed also on Macrophages, the most likely cell type will be assigned based on the combination of provided markers for CD4 T cells (CD3, CD4) and macrophages (CD68, CD163, CD206). However, for the subtype definitions, if CD4 is included only for CD4 T cells, the cell subtype assignment considers this marker to be specific for CD4 T cells, and renders any $CD4^+$ macrophage as Ambiguous. In addition, any co-expresion due to non-specific antibody binding also needs to be considered in this step. Therefore, the subtype definitions step requires understanding of the specificity for the subtype-specific markers in the context of the analysed type of tissue and experimental settings (**Supplementary Methods**).

Marker selection for positivity detection

TYPEx determines marker positivity based on three markers and their co-expression patterns. By default, these are CD3, CD4 and CD8a. If these three markers are not present in the cohort, the implementation of TYPEx allows for any markers with similar patterns of co-expression and mutual exclusivity to be specified in the input config file, *typing_params.json* (**Supplementary Methods** in Magness *et al.*).

Note, any markers that meet the criteria to define rare and frequent subpopulations can be used instead of the default. For example, if CD3 is not included in the cohort, CD45 can be used to replace CD3, where the frequent subpopulations can be defined as $CD45^+CD4^+$ and $CD45^+CD8^+$ whereas rare subpopulations can be $CD4^+$ ($CD45^-$), $CD8^+$ ($CD45^-$), $CD4^+CD8^+$, and $CD45^+CD4^+CD8^+$.

An important consideration is also any non-specific binding the default T cell markers may have. In the Barretts's esophagus CODEX dataset, we observed high CD8 intensities on the myeloid cell populations (CD11b+ and CD15+). Therefore, the input config file *typing params json* was modified so that single-positive CD8 was excluded from the list of rare subpopulations and single-positive CD4 was added instead. This model with CD8 added to the Major cell lineage took longer compared to the HuBMAP dataset, which has a comparable number of cells and major cell lineage definitions (**Extended Data Table 1**).

Tissue segmentation and tissue domain analysis

To permit quantification of cell counts per tissue area, a random-forest classifier is applied using selected channel images (**Fig. 3e**, **Supplementary Table S5** in Magness *et al.*). Alternatively, tissue annotations can be provided to the TYPEx module as input binary masks. TYPEx allows the assignment of cells to domains derived through any independent segmentation or manual labelling approaches, such as tissue structures like vessels or airways, or derived tissue domains such as adjacent-to-tumour. TYPEx then outputs the cell objects and their identities with the localisation information within these domains automatically.

Pathology review of H&E stain taken serially to the study sections prior to data acquisition is a valuable practice in order to select regions of interest for scanning in larger tissue sections, avoid unnecessary scanning of poor quality TMA cores and record histological features that may hold biological relevance. Using the TRACERx IMC datasets, we generated binary masks of pathologist annotations directly on IMC images through the use of the pseudo-H&Es output by deep-imcyto, in conjunction with conventional H&E images of serial TMA sections (**Extended Data Fig. 4** in Magness *et al.*). We found this particularly useful for the identification of alveolar macrophages, necrosis, large vessels, as well as the distinction of tumour and normal epithelial tissue in TMA cores composed of both tumour and normal epithelial cells.

Similarly, TYPEx can be configured to consider the binary masks as areas for exclusion. This feature can be useful when there are relatively large areas that have been identified as technical artefacts. The cell objects within these areas will be excluded from the cell phenotyping analysis.

Workflow options in Spatial-PHLEX

The primary aim of Spatial-PHLEX is to make quantitative measurements of tissue structures and derive measurements of cell spatial conformations and interactions which may occur over differing length scales. It does this through two means: spatial clustering and barrier scoring.

Spatial clustering

Given an input dataframe of cell objects this workflow performs spatial clustering for all unique cell types identified within the specified *phenotyping_column,* or alternatively for a single specified cell type. The workflow calculates the membership of all other cells to the identified clusters and calculates the distance of all cells to the nearest cluster boundary. For all clusters that are

identified, Spatial-PHLEX calculates the intracluster densities and proportions of all other cell types.

Barrier scoring

The barrier scoring workflow constructs a nearest spatial neighbours graph of the cells in an image, measures the shortest paths from all instances of cell type A to cell type B, and quantifies the presence of a cell type C along these paths in a variety of ways (**Box 1**).

Clustered barrier scoring

The clustered variant on the barrier scoring workflow constructs a nearest-spatial-neighbours graph of the cells in an image, measures the shortest paths from all instances of cell type A to clusters of cell type B, and quantifies the presence of a cell type C along these paths.

Barrier scoring metrics

The Spatial-PHLEX module allows the user to select a trio of cell types on which to perform barrier scoring, interrogating the extent to which a barrier cell type is spatially inter-positioned between a second and third cell type, the latter clustered by the DB-Scan spatial clustering method also invoked in Spatial-PHLEX. Six different barrier scores are automatically output by Spatial-PHLEX. Using the example of a fibroblast cell type as the barrier cell type between CD8 T cells and tumour cell spatial clusters, each metric responds to a slightly different question -

Binary barrier - How often is a fibroblast spatially inter-positioned between a CD8 T cell and tumour nest on at least one path from CD8 T cell to tumour nest?

Binary adjacent barrier - How often is a fibroblast spatially inter-positioned between a CD8 T cell and tumour nest and positioned at the tumour-stroma interface, on at least one path from CD8 T cell to tumour nest?

Weighted barrier - To what extent are fibroblasts spatially inter-positioned between CD8 T cells and tumour and positioned in the vicinity of the tumour nest?

All-paths barrier fraction - How often is a fibroblast spatially inter-positioned between a CD8 T cell and tumour nest, accounting for all possible routes from CD8 T cell to tumour nest?

All-paths adjacent barrier fraction - How often is a fibroblast spatially inter-positioned between a CD8 T cell and tumour nest and positioned at the tumour-stroma interface, accounting for all possible routes from CD8 T cell to tumour nest?

Barrier content - How many fibroblasts are typically spatially inter-positioned between a CD8 T cell and tumour nest?

Computational infrastructure

Hardware requirements

TRACERx-PHLEX was developed and tested on Linux-based institutional cluster running SLURM and SGE task schedulers. It is designed for reproducible analysis of large-scale IMC imaging projects, and hence is best run on a high-performance computing system (HPC). However, thanks to its Nextflow implementation it is adaptable to many different compute infrastructures.

Deep-imcyto

Deep-imcyto requires a system with both CPU and GPU computing resources available. In cluster environments deep-imcyto is configured by default to increase resource requests of CPUs, memory and time, if a job fails due to insufficient resources being allocated. Deep-imcyto CPU-based processes require a minimum of 16GB memory, and GPU processes also require a GPU with at least 16GB GPU memory available.

TYPEx

TYPEx uses up to 4 CPUs and 6 GB for the test dataset. These requirements vary based on the number of cells and number of major cell lineages, which can be specified in the config file provided with the argument *-c*. For the entire TRACERx dataset on T cells & Stroma panel with 3.16 million cells, maximum 25 CPUs and 98 GB were required.

Spatial-PHLEX

To run the default (clustered barrier) or barrier workflow of Spatial-PHLEX requires a system with both CPU and GPU computing resources available. In cluster environments Spatial-PHLEX is

configured by default to increase resource requests of CPUs, memory and time, if a job fails due to insufficient resources being allocated. Spatial-PHLEX requires only CPU resources for the spatial clustering workflow. CPU-based processes require a minimum of 16GB memory, and GPU processes also require a GPU with at least 16GB GPU memory available.

Software requirements

TRACERx-PHLEX has been tested using Nextflow/22.04.0 and Singularity/3.6.4. All other software needed to run TRACERx-PHLEX is distributed as Docker containers that are automatically downloaded and built by the PHLEX submodules when the pipeline is run. TRACERx-PHLEX is open source and available at https://github.com/FrancisCrickInstitute/TRACERx-PHLEX.

All software requirements have been packaged in executable Docker containers.

To follow the steps in this tutorial the user will also need access to a text or code editor for script editing.

Test dataset

The test dataset consists of five regions of interest from the TRACERx IMC dataset. These regions were selected to represent the range of cell subtypes detected in the wider TRACERx 100 cohort with the T cell & Stroma panel. The cores represented samples from different tissue and histology types: lung squamous cell carcinoma (LUSC), lung adenocarcinoma (LUAD) with acinar growth pattern, LUAD with solid growth pattern, pleomorphic carcinoma and a lymph node tissue sample. The LUSC core included tumour cells with wide-ranging pan-cytokeratin intensities and an area with densely packed T cells. The LUAD sample with solid growth pattern included non-nucleated αSMA^{+} fibroblasts that were captured with deep-imcyto in MCCS mode for cell segmentation. The LUAD sample with acinar growth pattern was representative of a high fibroblast barrier score. Most tumours cells in the sample from pleomorphic carcinoma had acquired mesenchymal properties. The lymph node core included a densely packed immune region without tumour content. The test dataset consists of five files in .ome.tiff format used as input for the TRACERx-PHLEX pipeline.

TRACERx-PHLEX Setup

Installation

!CRITICAL STEP: Ensure the necessary software is installed on your system. Using the TRACERx-PHLEX pipeline requires the following:

- Git
- Nextflow
- Singularity/Docker

!CRITICAL STEP: Find the nf-core profile for your computing environment. Setting the nf-core profile should allow PHLEX to run on your system without further modification. You will specify this profile later when creating scripts to run parts of PHLEX using the *-c* argument. See **Tutorial 1 - Step 4: Prepare the deep-imcyto shell script** for an example. See https://github.com/nf-core/configs for currently available institutional profiles. If an nf-core profile is unavailable for your institution, contact your administrator or follow the instructions here https://nf-co.re/docs/usage/tutorials/step_by_step_institutional_profile.

Download TRACERx-PHLEX and the PHLEX test dataset

Step 1: First create a suitable directory for testing PHLEX.

```
$ mkdir PHLEX_testing
$ cd PHLEX_testing
```

Then clone TRACERx-PHLEX from GitHub.

```
git clone --recursive git@github.com:FrancisCrickInstitute/TRACERx-PHLEX.git
```

!CRITICAL STEP: TRACERx-PHLEX consists of git submodules, so ensure that the –recursive flag is specified when you clone the repo.

Step 2: Load Nextflow on your system and launch the TRACERx-PHLEX setup pipeline, which will download all the necessary prerequisites for running PHLEX, including neural network weights and test images. A quick start guide is available for new users at https://tracerx-phlex.readthedocs.io/en/main/quickstart.html.

```
nextflow run TRACERx-PHLEX/PHLEX_setup.nf -w scratch
```

If you choose not to run the PHLEX setup pipeline, then the PHLEX test images, deep-imcyto neural network weights, and the TRACERx Nuclear IMC Segmentation Dataset can be downloaded directly from Zenodo at the following link: https://doi.org/10.5281/zenodo.7573268.

TRACERx-PHLEX Step-by-step tutorial

Tutorial 1: Segmenting cells in IMC images with the deep-imcyto module

Timing: 0-30 minutes per ROI

Input data

deep-imcyto supports unprocessed IMC data in the following formats:

- .mcd
- .txt
- multi-channel .tiff / .ome.tiff

The PHLEX testing dataset consists of 5 IMC images in .ome.tiff format.

!CRITICAL STEP: Ensure your marker and metal names are consistent across your input images. deep-imcyto will derive measurements based on the channel names provided in the input files. If these are not consistent then the discordant labels will be considered as different marker-metal pairs, which will require correction prior to cell typing analysis.

Step 1: Choose the workflow for deep-imcyto to execute

Parameters controlling how deep-imcyto executes can be set from the command line when deep-imcyto is run (See **Prepare the deep-imcyto shell script** for usage). The primary consideration for deep-imcyto is which workflow should be executed, which is dictated by experimental stage and user requirements (See **Experimental design**). The following options are available: ‘QC’, ‘simple’ and ‘MCCS’.

Step 2: Create the deep-imcyto metadata file

Timing: 0-5 minutes

The metadata file is a comma-delimited file containing information about the isotope channels in the input image files. An example metadata file is distributed in the TRACERx-PHLEX GitHub repository, and the file format shown in **Table 1**.

The first column is the name of the isotope channel (e.g. 141Pr). This should be the same as the name of the corresponding channel in the input image files. The subsequent columns contain binary values and represent whether a particular IMC isotope channel image should be used in a given set of deep-imcyto's processes. This can be one of the following:

- full_stack - The isotope channels will be used in the full-stack workflow. These are all the isotopes of interest in the experiment being performed.
- mccs_stack - The isotope channels to be used in the MCCS workflow. These will be dependent on how the MCCS workflow is implemented in CellProfiler (see **Supplementary Information**).
- nuclear - The isotope channels to be used to construct the nuclear image for nuclear segmentation processes. Note: a maximum of two channels should be specified.
- spillover - The isotope channels to which spillover compensation should be applied in the spillover correction workflow.

  !CAUTION: The number of spillover channels specified must be exactly equal to the number of rows/columns in the spillover matrix.

- counterstain - The isotope channels will be used to create the counterstain image to generate pseudo-H&Es. An intensity projection of these channels will be used to create the 'pseudo-counterstain'; we recommend channels not specified in the 'nuclear' column.

!CRITICAL STEP: Ensure correct specification of IMC panel isotopes in the metadata file for each workflow, otherwise deep-imcyto may fail to run correctly.

metal	full_stack	mccs_stack	nuclear	spillover	counterstain
80ArAr	1	0	0	0	0

100Ru	1	0	0	0	0
131Xe	1	0	0	0	0

Table 1: The deep-imcyto metadata file format.

Step 3: Set other parameters for workflow execution

Timing: 0-5 minutes

A full description of all deep-imcyto parameters is given in the deep-imcyto documentation on readthedocs (**Documentation**). Critical parameters that should be set by the user in all workflows are provided in **Table 2**. Other parameters that should be set for specific workflows are given in **Table 3**.

parameter	**description**
input	Path to an input file or input files. Can include glob/wildcard patterns e.g. /path/to/data/*.mcd
outdir	Path to the output directory.
release	A useful identifier for the results release.
nuclear_weights_directory	The path on the system to the directory where the deep-imcyto neural network weights are saved.
segmentation_workflow	‘QC’, ‘simple’, ‘CellProfiler’ or ‘MCCS’
metadata	Path to the deep-imcyto metadata file.

Table 2: Critical parameters in deep-imcyto.

Workflow-specific critical parameters are given below.

parameter	**description**	**Workflows**
full_stack_cppipe, mccs_stack_cppipe, segmentation_cppipe	.cppipe files necessary for an MCCS implementation*	MCCS, QC
dilation_radius	Amount to expand nuclear	simple

	predictions by in the simple workflow (default: 1px).	
simple_preprocess_method	Preprocessing method to use in simple workflow ('hotpixel', 'cellprofiler', 'none')	simple
qc_preprocess_method	Preprocessing method to use in QC workflow ('hotpixel', 'cellprofiler', 'none')	qc

Table 3: Key workflow specific parameters in deep-imcyto.

*See **Tutorial 4: Designing an MCCS workflow in CellProfiler**

Step 4: Prepare the deep-imcyto shell script

Timing: 0-5 minutes

Use a text editor to create a shell script file which will be used to launch the deep-imcyto Nextflow pipeline. To do this first create an empty text file, then copy in one of the examples below, depending which workflow you wish to run. Edit the relevant file paths (e.g. for `nuclear_weights_directory`) specified in the `nextflow run` command so that they correspond to the correct locations on your file system. Save this file with an appropriate filename e.g. *`run_deepimcyto.sh`*.

Simple segmentation

```
# Define folder for deep-imcyto software containers to be stored:
export NXF_SINGULARITY_CACHEDIR=".singularity"

# RUN DEEP-IMCYTO in MCCS mode:
nextflow run TRACERx-PHLEX/deep-imcyto/main.nf\
    --input "./data/*.mcd"\
    --outdir "../results"\
    --release "PHLEX_example"\
    --nuclear_weights_directory "./deep-imcyto_weights/"\
    --segmentation_workflow "simple"\
    --metadata "./metadata/deep-imcyto_metadata.csv"\
    -profile {your_institution_nf-core_profile}\
    -w "./scratch"\
    -resume
```

CellProfiler mode using the MCCS workflow

```
nextflow run TRACERx-PHLEX/deep-imcyto/main.nf \
   --input "./data/*.mcd"\
   --outdir "../results"\
   --release "PHLEX_example"\
   --metadata "./metadata/deep-imcyto_metadata.csv"\
   --nuclear_weights_directory "./deep-imcyto_weights/" \
   --segmentation_workflow 'MCCS' \
   --full_stack_cppipe
"./assets/cppipes/MCCS/full_stack_preprocessing.cppipe"\
   --segmentation_cppipe "./assets/cppipes/MCCS/segmentationP1.cppipe" \
   --mccs_stack_cppipe
"./assets/cppipes/MCCS/mccs_stack_preprocessing.cppipe" \
   --compensation_tiff "./assets/spillover/compensation.tiff" \
   --plugins "./assets/plugins" \
   -profile {your_institution_nf-core_profile} \
   -w './scratch' \
   -resume
```

!CRITICAL STEP: Replace {your_institution_nf-core_profile} with the appropriate nf-core profile for your system (See **Installation**).

!CRITICAL STEP: Ensure you have the necessary system permissions in the NXF_SINGULARITY_CACHEDIR, or singularity containers may fail to download, build or execute during running of deep-imcyto.

Step 5: Run deep-imcyto

Launch the deep-imcyto module by running the script from the command line:

```
$ ./run_deep-imcyto.sh
```

Deep-imcyto will begin processing the input images and segmentation outputs will be generated.

Troubleshooting of deep-imcyto

Troubleshooting advice for running deep-imcyto can be found at https://tracerx-phlex.readthedocs.io/en/main/deep-imcyto.html#troubleshooting.

deep-imcyto will attempt to mount all file paths inside the necessary singularity containers automatically, however if this does not work (e.g. the PHLEX test images or deep-imcyto weights are not able to be found on the filesystem when deep-imcyto tries to read them) then the user may need to manually specify the locations on the host system that should be bound to the container. This can be done with the following command line option. Multiple locations can be specified separated by a comma:

```
    --singularity_bind_path='/camp,/nemo'\
```

See the Singularity documentation for more detailed advice [10].

Deep-imcyto may occasionally fail to run due to the quality of the input images. For instance, if nuclear staining quality is extremely poor then the pipeline may detect zero nuclei in the input images. If this does occur deep-imcyto's raw channel output, preprocessed nuclei, and pseudo-H&E images can then be inspected for poor nuclear signal, which may be due to poor sample quality, necrosis or poor tissue preservation, for instance.

Missing channel images, pseudo-H&E, nuclear preprocessing images. This may be due to incorrect specification of the metal isotope tags in the deep-imcyto metadata file. Ensure the metal tags are correctly specified for your experiment in the deep-imcyto metadata file and try again.

Timing for deep-imcyto

Execution timings for deep-imcyto depend on the workflow, the type and size of the images and the computing resources available. If running the CellProfiler workflow with a custom CellProfiler .cppipe then the time will depend on the processes specified within CellProfiler.

For our CellProfiler MCCS implementation, when applied to 277 TRACERx 100 IMC images the mean processing time per ROI was 29.3 minutes. This largely reflects the size of TRACERx 100 images, the mean number of pixels in a TRACERx 100 image being approximately 9.3x larger than the Jackson et al breast cancer IMC cohort. Example processing times for individual processes in deep-imcyto's workflows applied to the PHLEX test dataset are provided in **Table 4**.

We ran the deep-imcyto simple segmentation workflow on 746 ROIs from Jackson *et al*. with a mean execution time per ROI of 2.0 minutes. Provided the user has sufficient compute resources

available on their system, deep-imcyto performs parallel processing of individual images so that even a large cohort can be processed in a reasonable time.

Workflow	Process	Process description	Execution time (minutes)
simple	IMCTOOLS	Format IMC image data for downstream processes	0.3
simple	CELL_MEASURE	Make measurements of intensity and morphometry of cells	2.9
simple	NUCLEAR_MEASURE	Make measurements of intensity and morphometry of nuclei	2.5
simple	NUCLEAR_DILATION	Dilate nuclear masks	0.1
simple	NUCLEAR_PREPROCESS	Preprocess IMC nuclear channels for deep-learning prediction	0.1
simple	NUCLEAR_SEGMENTATION	Segment nuclei with the deep-imcyto model	2.7
simple	OVERLAYS	Make overlay plots of segmentation masks	0.2
simple	PSEUDO_HE	Make pseudo H&E images from IMC channels	0.4
MCCS	MCCS	Perform MCCS via Cellprofiler	13.2
MCCS	IMCTOOLS	Format IMC image data for downstream processes	0.4
MCCS	NUCLEAR_PREPROCESS	Preprocess IMC nuclear channels for deep-learning prediction	0.2
MCCS	NUCLEAR_SEGMENTATION	Segment nuclei with the deep-imcyto model	2.8
MCCS	PREPROCESS_FULL_STACK	Preprocess all IMC channels for measurement	4
MCCS	PREPROCESS_MCCS_STACK	Preprocess channels to be used in MCCS via Cellprofiler	4.3
MCCS	PSEUDO_HE	Make pseudo H&E images from IMC channels	0.4

Table 4: Recorded deep-imcyto processing times for the PHLEX test dataset.

Anticipated results for deep-imcyto

Workflow 1: QC

When run on a raw .mcd, .ome.tiff or other input file in QC mode, the user can expect the following outputs:

1. Raw single channel .tiff images corresponding to the metals specified in the deep-imcyto metadata file.

2. Preprocessed versions of the above tiffs with the preprocessing steps specified by the user.
3. Independently contrast-adjusted versions of the raw image channels, useful for visualisation and manual review.

Workflow 2 & Workflow 3: simple segmentation and CellProfiler

Fig. 9a-b gives an illustrative overview of the outputs expected for a given sample from deep-imcyto in *simple* segmentation mode. These are:

1. Segmentation masks
2. Single channel .tiffs files (raw and preprocessed)
3. Pseudo-H&E images for each ROI
4. Marker intensity and morphometric measurements of every nucleus and cell (as *nuclei.csv* and *cells.csv* respectively)
5. Nearest neighbour measurements of each identified object
6. Spatial plots showing the spatial distribution of the mean and standard deviation of marker intensity per cell of each marker (each marker is independently normalised to allow for visualisation)

The outputs obtained from CellProfiler mode are variable and depend on the user's specification of the processes in the CellProfiler pipeline file (for an example of the processes we implemented for our MCCS protocol see **Tutorial 4: Designing an MCCS workflow in CellProfiler)**.

Tutorial 2: Cell phenotyping with TYPEx

1 Provide input cell-by-marker table

A) When using input from deep-imcyto

1. Specify the path to the deep-imcyto's release folder by adding the argument *–input_dir* to the Nextflow command that launches TYPEx
2. Set the argument *–deep_imcyto* to the value true
3. Indicate whether the segmentation was performed in CellProfiler (true) or *simple* mode (false)

```
nextflow run TRACERx-PHLEX/TYPEx/main.nf \
```

```
            --input_dir $PWD/results/deep-imcyto/$release/ \
            --deep_imcyto true -cellprofiler true
```

B) In standalone mode

1. Provide a tab-separated cell objects table with the following columns in the defined order:

ObjectNumber	imagename	X	Y	Area	<Marker 1>	…	<Marker N>

2. Specify the path to the cell objects table using the argument *–input_table*

```
nextflow run TRACERx-PHLEX/TYPEx/main.nf \
            –input_table $PWD/TYPEx/data/cell_objects.tracerx.txt
            --deep_imcyto false
```

!CRITICAL STEP: TYPEx accepts only non-negative definite values for the marker intensities in the cell object table. The range of values can be rescaled by adding the minimum value to all intensities. In addition, the rows with NAs will be excluded.

!CRITICAL STEP: When used as a standalone module, TYPEx assumes that the upstream steps account for processing requirements specific to the imaging modality, such as removing noise, and optical artefacts.

Setting up the cell-type definitions file to the panel of profiled markers

2 Customise the definitions of major cell-lineages to the antibody panel

TYPEx creates a probabilistic model that annotates each individual cell, based on the major cell lineage definitions provided as input config file (*cell_type_annotations.json*). Start with a template file provided with TYPEx distribution. Add or remove major cell lineages and defining markers based on the list of profiled markers in the panel. An example of major cell lineage definitions is provided in **Figure 1c**.

3 Define the cell subtypes with the subtype-specific combination of positive markers.

Start with a template file provided with TYPEx distribution. Add or remove cell subtypes and defining markers based on the list of profiled markers in the panel. Consider whether the subtype-defining markers are expressed by multiple cell subtypes or have non-specific staining. Refer to

the **Experimental design** section or the TYPEx documentation for further advice on how to define cell subtypes: https://tracerx-phlex.readthedocs.io/en/main/TYPEx.html#guide.

Creating the table with sample information

4 Create a tab-delimited file with a list of image names in the dataset.

In the imagename column, include the unique identifiers for each image.

When using deep-imcyto, the unique image identifier is determined based on the input raw image names, in the format *{mcd}-{roi}* for mcd inputs or {tiff_filename}-{lowercase_tiff_filename} for tiff, ome.tiff and text inputs.

In standalone mode, the *imagename* values are required to match the *imagename* column in the input cell objects file.

imagename	<experimental condition>	<Batch effect 1>	...	<Batch effect N>	use_image

Optionally, add column(s) that can be considered for batch effect correction or add column(s) with sample information, *e.g.* experimental condition and clinical information to be added in the summary cell density table.

To analyse only a subset of images, add the column use_image and add the values “exclude” for the images selected for exclusion.

Tuning TYPEx typing parameters

5 Specify the markers for calling positivity.

In the t*yping_params.json*, by default, marker positivity is determined based on the co-expression patterns of the three T cell markers, CD3, CD4 and CD8. To check whether these markers are present in the antibody panel, check whether they match the column names in the input cell objects table provided in standalone mode or the cells.csv table generated from deep-imcyto.

If these three markers are not present in the cohort, the implementation of TYPEx allows for any markers with similar patterns of co-expression and mutual exclusivity to be specified in the input config file, typing_params.json.

```
"threshold":{
      "markers":["CD3", "CD4", "CD8"],
        "high_frequency":["CD3_CD4", "CD3_CD8"],
        "variable":["CD4"],
        "low_frequency":["CD3"],
        "rare":["CD8", "CD4_CD8", "CD3_CD4_CD8"]
  },
```

!CRITICAL: An important consideration is also any non-specific binding the default T cell markers may have (**Supplementary Methods**).

6 Specify the populations of markers for calling positivity.

Skip this step if all three markers, CD3, CD4 and CD8 are present in the panel.

TYPEx estimates an optimal D-score threshold that minimises the populations listed as "`rare`" but maximises those listed "`high_frequency`". By default, in the config file, we specify that the double-positive $CD3^{+/-}CD4^{+}CD8a^{+}$ and single-positive $CD8a^{+}$ cells are expected to be rarely found in peripheral non-lymphoid tissue, whereas $CD3^{+}CD4^{+}$ and $CD3^{+}CD8a^{+}$ to be the high_frequency populations in the analysed cohort. If different markers are specified, include the combinations of positive markers using underscore to separate the markers and use the same order in which they are specified in the "`markers`" list. The "`variable`" population is relevant because TYPEx will expect to find a non-zero count for this population (**Fig. 3c-d** in Magess *et al.*). Metals names are not included in the list of markers.

7 Include batch effects and experimental condition [OPTIONAL]

In the *typing_params.json*, select the column names in the sample annotation file that should be considered as batch effects and list these in the `batch_effects` list. Similarly for the columns to be added in the cell density table, include these columns in the list `experimental_condition`.

8 Specify minimum intensity value [OPTIONAL].

A minimum threshold for marker intensity can be set to consider only cells that have a higher intensity than the specified threshold for at least one major cell-lineage marker.

```
 "min_expresssion":[0],
```

9 Exclude channel measurements [OPTIONAL]

Exclude any measurements not relevant for phenotyping by specifying the corresponding isotope prefix or marker name as follows

```
"channels_exclude":["80ArAr"],
```

10 Set the magnitude parameter [OPTIONAL].

The raw marker intensities are multiplied by a factor of ten for building the probabilistic model with CellAssign. Change the default `magnitude` value from 10 if the input intensities have been rescaled. Specify the magnitude parameter when using MCCS mode to 10^6 and when the intensity range has been rescaled in the range 0-1 (**Supplementary Methods** in Magess *et al.*).

Tissue segmentation settings

11 Include the Epithelial and stroma-specific list of markers for tissue segmentation [OPTIONAL] .

Include the names of the images for the corresponding tissue category. By default, TYPEx looks for these images in `<input_dir>/imctools/<file_id>/<roi>/full_stack/*` both when using deep-imcyto or in standalone mode. The list `auxstroma` is optional and can be used when there are stromal markers such as vimentin, collagen or panactin that define the stromal areas in addition to other stromal markers but can also be expressed in the tumour tissue category.

```
"markers":{
      "tumour":"164Dy_panCK.tiff",

      "immune":"152Sm_CD45,170Er_CD3.tiff,162Dy_CD8a.tiff,156Gd_CD4.tiff",
      "stroma":"141Pr_aSMA.tiff,151Eu_CD31.tiff",
      "auxstroma":"143Nd_vimentin.tiff,169Tm_collagen1.tiff",
      "dna":"191Ir_DNA1.tiff,193Ir_DNA2.tiff"
}
```

12 Provide binary masks for annotations of tissue domains [OPTIONAL]

Provide pathology annotations or segmented tissue masks independently of deep-imcyto in the following format. Specify the value for annotationName as the one that will be used as column name in the cell objects file. Indicate regular expressions that will be used to extract the unique image identifier from the tiff file name to be matched to the cell objects table. Optionally, a sample or patient identifier can be extracted with `regionRegExIndex`.

```
"masks":{
            "tisseg":{
                  "tissueDir":"path/to/tissueDir/",
                  "annotationName":"regional",
                  "maskRegEx":"labels_(region.*)_(image_.*).tiff",
                  "imgNameRegExIndex":"\\2",
                  "regionRegExIndex":"\\1"
            }
}
```

13 Artefact masks for area exclusion [OPTIONAL].

In cases of large areas with artefacts, the user can provide a folder with masks for those areas as input, and the cell objects within these areas will be excluded from the cell phenotyping analyses. Note, the value for `annotationName` must be "*exclude*".

In certain cases there can be large and frequent areas with antibody aggregates or artefacts in the dataset, the marker intensities can be among the highest in the cohort in these areas. To ensure that these artefacts may affect results, masking out such areas by providing the artefacts masks as input for TYPEx will resolve any issues.

```
"masks":{
      "exclude_mask":{
            "tissueDir":"/path/to/dir/",
            "annotationName":"exclude",
            "maskRegEx":"(image.*)_(region.*).tif",
            "imgNameRegExIndex":"\\1",
            "regionRegExIndex":"\\2"
      }
}
```

Set the workflow parameters for running TYPEx

14 Provide the sample annotation file

Specify the sample annotation file created in step number 4, using the argument *–sample_file*.

15 Include config files

Indicate the path to the config files adjusted in the previous steps, typing `params config`, `annotation config`, tissue segmentation `overlay config`, `colour config` for visualisation. If these arguments are not specified, the default paths point to the templates within the TYPEx distribution.

```
  –-params_config "$PWD/config/typing_params.json" \
  --annotation_config "$PWD/config/cell_type_annotation.json" \
  --overlay_config_file "$PWD/config/tissue_segmentation.json" \
  --color_config "$PWD/config/celltype_colors.json"
```

16 Output settings [OPTIONAL]

Set the output folder and any name for the run specified with `––release`.

```
  --release $release \
  --output_dir "$PWD/results/TYPEx/$release/"
```

17 Tissue segmentation model [OPTIONAL]

We trained a three-class model for background, tumour nest/epithelium and stroma regions on images from tumour, adjacent-normal lung and lymph node tissue using Ilastik [11]. This model can be specified as

```
  --tissue_seg_model "$PWD/TYPEx/models/tumour_stroma_classifier.ilp"
```

To allow broader use across different tissue types, TYPEx also incorporates an alternative two-class model for tissue and background segmentation, which is distributed in the same folder *`tissue_classifier.ilp`*. Both use the single-channel files listed for each tissue category in the `tissue_segmentation.json` file.

Any classifier trained with Ilastik can be provided with the argument `—tissue_seg_model`, given that the training images are created in a similar way as composites of epithelial and stromal markers.

18 Launch TYPEx

Launch the TYPEx module by running the script from the command line

```
$ ./run_TYPEx.sh
```

Troubleshooting of TYPEx

Troubleshooting advice for running TYPEx can be found at https://tracerx-phlex.readthedocs.io/en/main/TYPEx.html#troubleshooting. A guide for configuring input parameters and further details for designing the cell type definitions table can be found at https://tracerx-phlex.readthedocs.io/en/main/TYPEx.html#guide.

Timing for TYPEx

The time and computational complexity for cell phenotyping can vary depending on the number of cells, the number of measured markers and the number of major cell lineages in the dataset. We tested TYPEx on images from both TMA cores and whole slides. The number of cells ranged from 140,000 to 3.2 million cells. The number of profiled markers in the antibody panels were 35 to 56. Finally, the number of different major cell lineages specified in the input model included between 8 and 24 lineage-specific markers.

The timing and computational requirements of individual processes within the TYPEx pipeline are demonstrated in **Table 5** for the test dataset. Detailed processing requirements for the datasets analysed in the study Magness *et al.* are included in **Table 6**. The most time intensive processes involve the building of the probabilistic models. Therefore, the outputs from these processes are saved and reused when TYPEx is rerun, for example, with different colour settings, or resumed after an interrupted run.

	process name	TRACERx test dataset **		
		duration	cpus	mem (GB)
Input formatting	format_input	3m	1	0.84
	collate_features		2	<1
Complete model	tier_one	1h 48m	4	5.1
Incomplete model	tier_one_ref	42m	4	4.2
Stratify by confidence	build_strata_model	2m	2	1
Positivity calling and assignment	tier_two	47m	1	1.9
Tissue segmentation	create_composites *	7m	2	6
	preprocess_panck *		2	3.3
	process_probs *		2	3.8
	run_classifier *		4	2.2
	ts_exporter *		1	<1
	mask_overlay *		1	<1
Export	subtypes_exporter	<1m	1	<1
QC and visualisation	qc_intensity_heatmap	2m	1	<1
	qc_select_images *	<1m	1	<1
	qc_create_single_channel_images *	1m	2	3.3
	qc_overlay *	4m	1	<1

* These steps are executed or produce outputs only when raw images are provided as input.

** Input cell objects derived from deep-imcyto in MCCS mode

Table 5. Timing, CPU and memory requirements of individual processes within TYPEx on the TRACERx test dataset.

Table 6. Timing of individual cell phenotyping processes with TYPEx across the datasets analysed in Magness *et al.*.

		TRACERx test dataset **			HUBMAP BE			HUBMAP colon			Schuerch *et al.*			TRACERx 100 (T cells & Stroma panel) **		
	process name	**duration**	**cpus**	**mem (GB)**	**duration**	**cpus**	**mem (GB)**	**duration**	**cpus**	**mem (GB)**	**duration**	**cpus**	**mem (GB)**	**duration**	**cpus**	**mem (GB)**
Input formatting	format_input	3m	1	0.84	2m	1	1.3	2m	1	1.5	2m	1	0.45	55m	2	3.5
	collate_features		2	<1		1	<1		1	<1		1	1.8		2	<1
Complete model	tier_one	1h 48m	4	5.1	3h 7m	27	30.4	2h 11m	27	43.8	14h 27m	27	37	5h 15m	25	98
Incomplete model	tier_one_ref	42m	4	4.2	2h 48m	27	26.8	2h 16m	27	39.9	8h 29m	20	35	4h 44m	20	78
Stratify by confidence	build_strata_model	2m	2	1	3m	2	1.3	2m	2	1.6	2m	2	1.5	26m	1	8
Positivity calling and assignment	tier_two	47m	1	1.9	1h 21m	1	2.8	1h 15m	1	3.9	1h 9m	1	3.6	1h 19m	1	14
Tissue segmentation	create_composites *	7m	2	6	<1m	3	<1	<1m	3	<1	<1m	2	<1	2h	2	7.2
	preprocess_panck *		2	3.3		2	1.5		2	<1		2	<1		1	3.8
	process_probs *		2	3.8		1	1.6		2	1		1	<1		2	4
	run_classifier *		4	2.2		1	<1		1	<1		1	<1		4	2.8
	ts_exporter *		1	<1		1	<1		1	<1		1	<1		1	<1
	mask_overlay *		1	<1		1	<1		1	<1		1	<1		1	<1
Export	subtypes_exporter		1	<1	<1m	1	<1	<1m	1	<1	1m	2	<1	2m	1	1.8
QC and visualisation	qc_intensity_heatmap	2m	1	<1	3m	2	<1	5m	2	1.1	2m	1	1.3	25m	1	8
	qc_select_images *	<1m	1	<1		/			/			/		<1m	2	3
	qc_create_single_channel_images *	1m	2	3.3										<1m	1	<1

	qc_overlay *	4m	1	<1				4m	1	<1

* these steps are executed or produce outputs only when raw images are provided as input. ** Input cell objects derived from deep-imcyto in MCCS mode

Anticipated results for TYPEx

Cell object and image visualisations

TYPEx outputs summary tables, plots and images to facilitate data interrogation, interpretation and quality control of the cell phenotyping results (**Table 7**).

Several images of pixel-level information and annotated single-cell objects are generated for visual inspection. Specifically, TYPEx outputs a map of cell objects coloured by the cell subtype for each analysed sample, as illustrated for three examples of the TRACERx test dataset (**Fig. 2a**). To visually assess the positivity and cell subtype calls, a pair of a raw single-channel image and the overlay of positive cell objects for a given marker are also generated (**Fig. 2b-c**). For each major cell lineage marker, the samples with the highest cell count for the corresponding major cell lineage are selected and visualised. The cell maps and overlays are generated when the mask of segmented cell objects is provided as input for TYPEx. Furthermore, the results from the tissue segmentation are output as raw composite images of epithelial- and stroma-specific markers and overlays of the tissue segmentation masks onto the composite image (**Fig. 2d**).

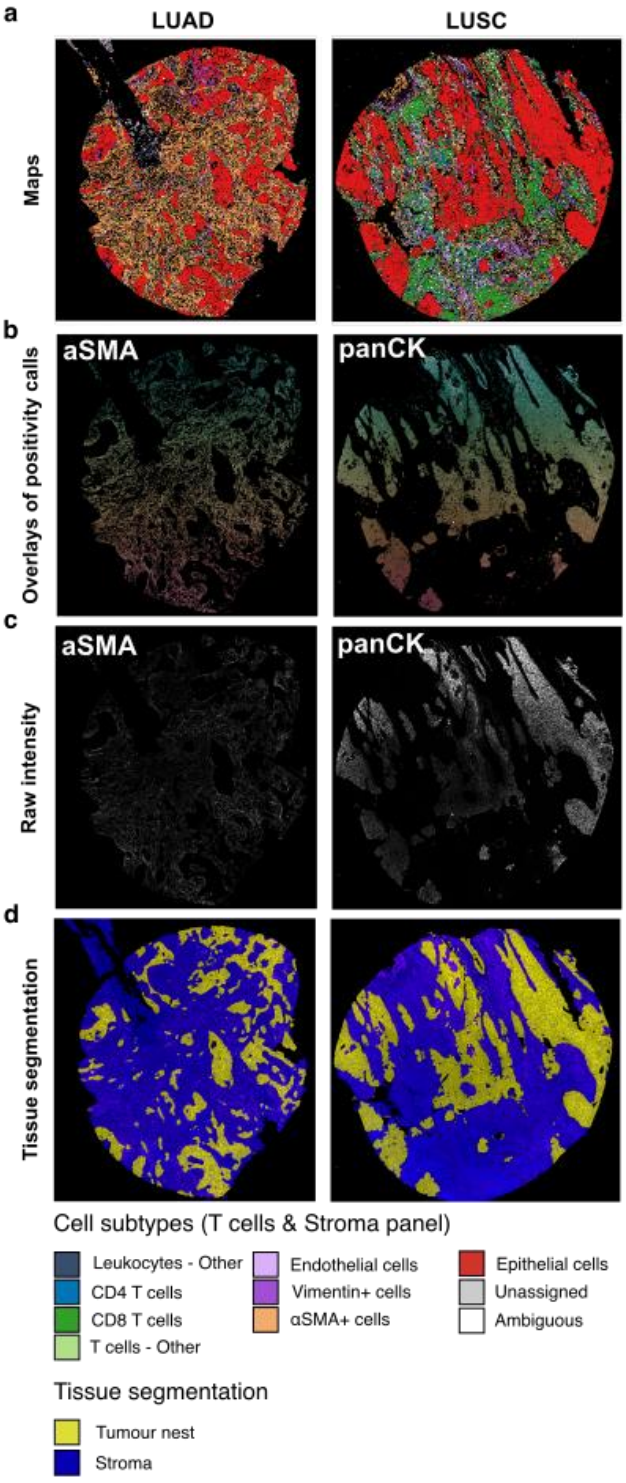

a
LUAD
LUSC
Maps
b
Overlays of positivity calls
aSMA
panCK
c
Raw intensity
aSMA
panCK
d
Tissue segmentation
Cell subtypes (T cells & Stroma panel)
Leukocytes - Other
CD4 T cells
CD8 T cells
T cells - Other
Endothelial cells
Vimentin+ cells
αSMA+ cells
Epithelial cells
Unassigned
Ambiguous
Tissue segmentation
Tumour nest
Stroma

Figure 2: Anticipated results for TYPEx - Example image outputs for three PHLEX test dataset cases.

TYPEx outputs various images for the user, shown here for three PHLEX test dataset cases (T cells & Stroma panel), including: **a**, A map of cell objects coloured by cell subtype for each analysed sample, **b**, The overlay of positive cell objects for a corresponding marker image. For each major cell lineage marker, the samples with the highest cell count for the corresponding major cell lineage are selected and visualised. **c**, A raw single-channel intensity image for a marker of interest. **d**, An overlay of tissue segmentation masks onto raw composite images of epithelial- and stroma-specific markers. The cell maps and overlays are generated when the mask of segmented cell objects is provided as input for TYPEx. LUAD, lung adenocarcinoma; LUSC, lung squamous cell carcinoma.

Summary figures for data exploration and applications in TRACERx

To allow for data exploration, TYPEx outputs a collection of heatmaps, cell frequency graphs and intensity plots (**Table 8**). The intensity distribution of each cell subtype-defining marker is visualised across all subtypes identified in the analysed dataset, separately for low and high confidence cell populations. For example, the intensities of the CD4 and CD8a markers are expected to be the highest for the CD4 and CD8 T cell populations, respectively. As shown for the T cells & Stroma panel in the TRACERx 100 IMC cohort (**Fig. 3a**), both the low and high confidence CD4 T cell populations have higher CD4 intensities relative to other cell subtypes within the same confidence group. Cells with high intensities of both CD4 and CD8a were labelled as $CD4^{+}CD8^{+}$ double-positive T cells (T cells DP).

In addition to cell subtype identification, TYPEx also determines the positivity status for each marker in the panel by automatically inferring a separation cutoff. To compare the intensity distributions between positive and negative cells for a given marker, TYPEx outputs intensity distribution plots per cell subtype and confidence group. This feature of TYPEx to detect marker positivity allowed a systematic analysis of subtype-specific checkpoint molecule expression in the TRACERx 100 cohort. For example, based on the ICOS intensities, the cells were automatically split into $ICOS^{+}$ and $ICOS^{-}$ (**Fig. 3b**). Based on this, we generated a catalogue of immune-checkpoint co-expression and the immune cells upon which they are expressed compared between the tumour nest and stroma tissue compartments in NSCLC (**Fig. 3c**).

When using TYPEx with the minimal input of a cell-by-marker intensity matrix and a sample annotation file without providing raw images, TYPEx generates a scatter map of cell objects illustrated as points and coloured by cell subtype. For example, using a region of healthy intestine from the HuBMAP CODEX data, the TYPEx annotations are visualised with spatial coordinates next to the ground truth data (**Fig. 4a**).

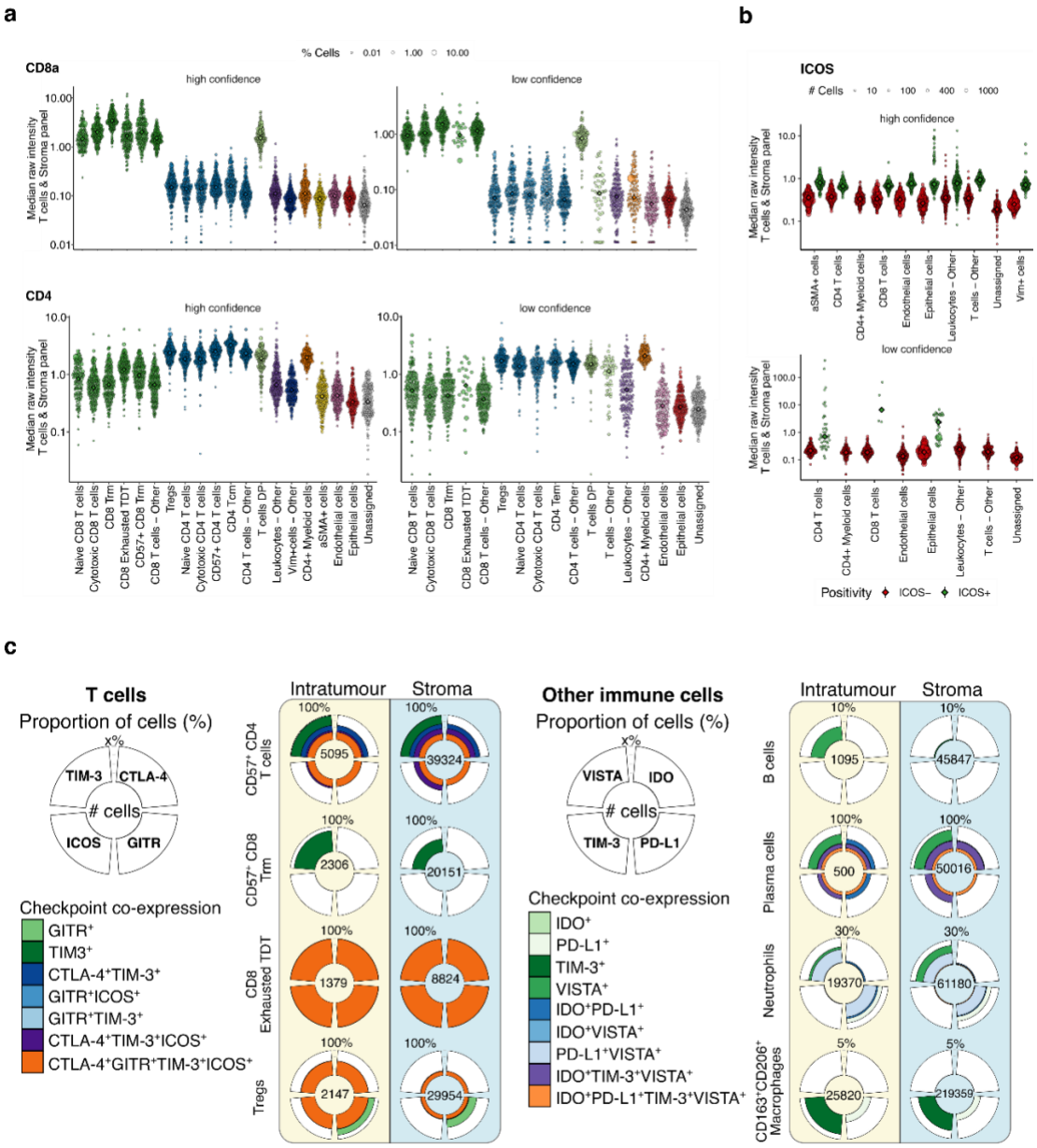


Figure 3: Anticipated results for TYPEx - Example intensity distributions and checkpoint molecule positivity calls across cell subtypes in the TRACERx 100 IMC cohort

a, The intensity distributions of subtype-defining markers CD8a and CD4 across all cell subtypes on the T cells & Stroma panel, split by confidence group. Point size shows the percentage of total cells in the cohort (T cells & Stroma panel). **b**, ICOS intensity distribution plots per major cell lineage and confidence group, split into $ICOS^+$ and $ICOS^-$ cells (T cells & Stroma panel). In **a**-**b**, each point represents an image, and measures the median of the cellular mean raw intensities of the relevant marker (y-axis). **c**, Catalogue of immune-checkpoint co-expression and the immune cells upon which they are expressed, compared between the tumour nest ('Intratumour') and Stroma tissue compartments in non-small cell lung cancer. T cells data is taken from the T cells & Stroma panel, and Other immune cells from the Pan-Immune panel. The percentage refers to the proportion of cells of the named subtype that express a given checkpoint molecule. Pie charts are shown with varying maximum cell proportions (e.g. 100% for T cells and 5% for $CD163^+CD206^+$ Macrophages) to minimise white space (where there is no checkpoint molecule expression) for better data visualisation. All data shown are for the TRACERx 100 IMC cohort. T cells DP, $CD4^+CD8^+$ double-positive T cells.

Summary figures for cell phenotypes and applications in HuBMAP

As the HuBMAP dataset had granular phenotype annotations, we used it to examine the combined information of marker positivity and cell subtypes. TYPEx generates a collection of heatmaps for data exploration and summary. First, TYPEx outputs a heatmap of the z-score normalised median intensity for all cell-subtype defining markers and identified cell subtypes in the dataset (**Fig. 4b**). For each cell object, a combination of positive markers is identified and, based on it, a cell subtype is automatically assigned. The cells are separated into multiple clusters and two confidence groups in the stratification step. Once the combination of positive markers is identified, the clusters with the same combination of positive markers and the same assigned cell subtype are considered the same cell subpopulation. For example, all clustered positive for CD68, CD163, CD206, and HLA-DR will be merged into one macrophage subpopulation. To illustrate all the unique combination of positive markers, TYPEx generates a heatmap for each cell subtype with the marker intensities and positivity calls for each subpopulation. For example, in the HUBMAP dataset, TYPEx identified 19 subpopulations of macrophages, distinguished by co-expression of macrophages markers with HLA-DR, CD16, CD4, CD11c and vimentin (**Fig. 4c**). Of note, some of the positive markers, such as CD3 in one macrophage subpopulation, can

represent signal spillover into neighbouring cells. Based on the positivity of other macrophage-specific markers, CD68, CD163, CD206, and macrophage-expressed markers, *e.g.* CD4, this subpopulation was assigned as macrophage rather than a T cell. Exploration of these heatmaps can also be useful to tune the cell subtype definitions to the specific dataset and rerun TYPEx, which resumes without repeating the time consuming processes such as the probabilistic model building.

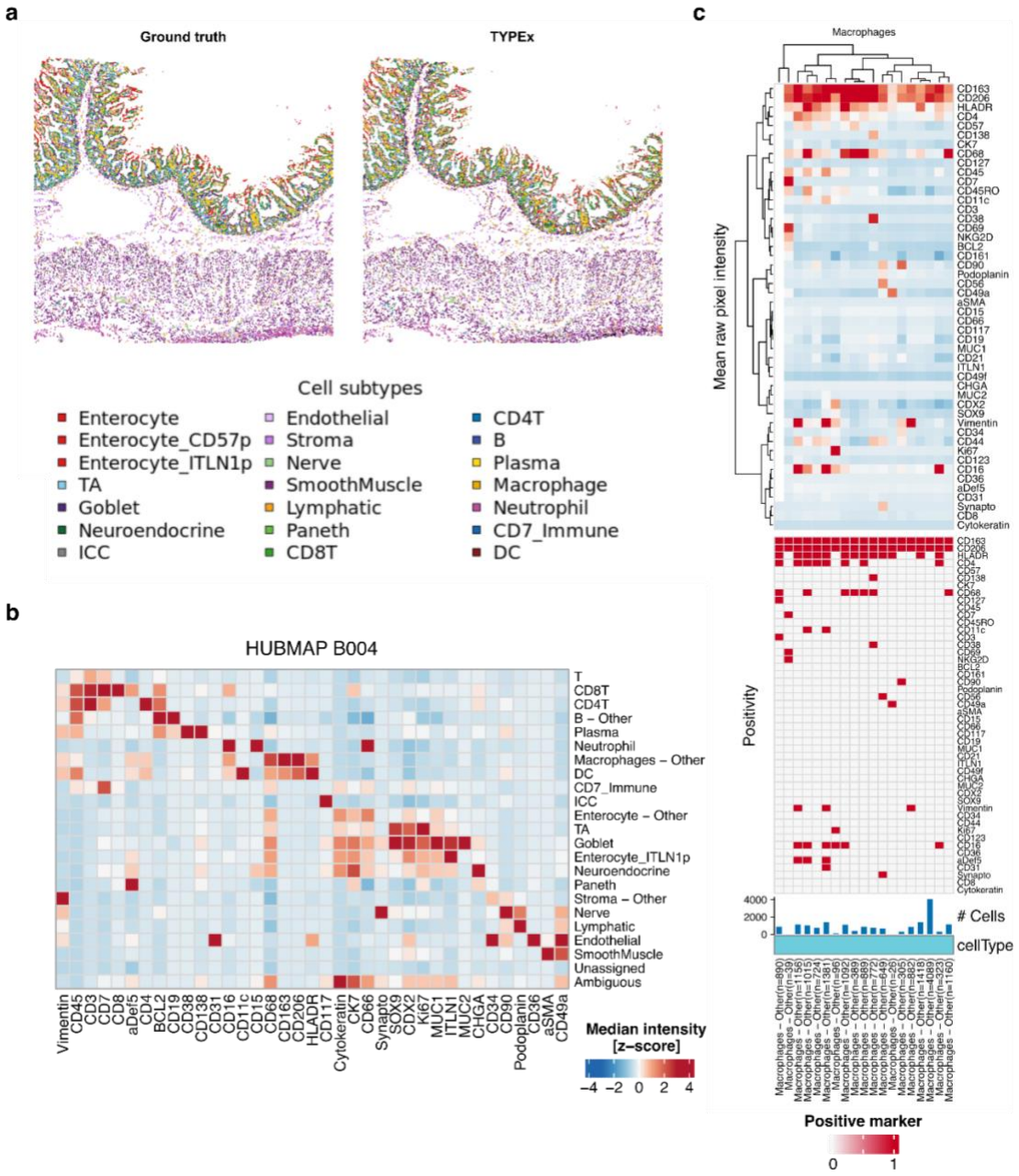


Figure 4: Anticipated results for TYPEx - Example outputs for HuBMAP CODEX data

a, TYPEx annotations visualised with spatial coordinates next to the ground truth data for a region of healthy intestine from the HuBMAP CODEX data for the scenario when raw images were not provided to TYPEx. **b**, Heatmap of the z-score normalised median intensities for all cell-subtype defining markers and identified cell subtypes in the dataset. **c**, TYPEx-generated heatmaps for each macrophage cell subtype identified in the HuBMAP dataset, showing marker intensities and positivity calls for each of 19 subpopulations distinguished by co-expression of macrophages markers with HLA-DR, CD16, CD4, CD11c and vimentin.

Summary tables for data analysis and quality control

Finally, TYPEx outputs several summary tables for interrogation of hypotheses (**Table 9**). First, a cell objects table reports the assigned cell subtype, tissue compartment, and combination of positive markers for each cell object in the analysed images. To allow for comparison of different samples, TYPEx performs normalisation to adjust for any differences in the area of imaged tissue between the samples. Therefore, cell counts are normalised per tissue area, derived from the tissue segmentation step. The cell density table summarises the cell density of each cell subtype per image, where the cell count is normalised by the segmented tissue area. If specified in the input typing params config file under experimental condition, additional columns such as clinical information and treatment will be added to this table for straightforward data analysis.

Two summary tables show the cell density of each cell subpopulation per cell subtype and image with the combination of markers that defines them; the first table reports the cell densities per total tissue area, whereas the second table reports the cell densities per tissue compartment. Additionally, a positivity table summarises the cell densities of single positive markers per image.

Finally, three tables can be useful for quality control of the TYPEx outputs. The table of T cells frequencies for each combination of positive T cells markers, CD3, CD4 and CD8a, can suggest suboptimal D-score cutoff when the proportion of double-positive CD8a and CD4 cells is higher than 10%. The table summarising the proportion of ambiguous and unassigned cells can also suggest a need to revise the major cell lineage or subtype definitions when the proportion of unassigned and ambiguous is higher than expected (for example, 10% when the panel targets all types of cells in the cohort). The phenotypes table shows every combination of positive markers in the dataset with the assigned cell subtype and confidence group.

Outputs	Path	Output file	Description	Figure
cell maps	**summary/*/maps**	types_*png	A map of cell objects coloured by the cell subtype annotation for each analysed sample, when the mask of segmented cell objects is provided as input.	Figure 1a
		scatter/*png	Spatially visualised single cells as points coloured by the assigned cell subtype.	Figure 4a
overlays	**summary/*/overlays**	*png	Raw intensity images of single channels adjusted for brightness and contrast.	Figure 1c
		outlines_*	Cell objects coloured by marker positivity overlaid onto the raw intensity image of the corresponding marker; output when the mask of segmented cell objects is provided as input.	Figure 1b
tissue segmentation	**composites**	composite images	Raw composite image merging channels from stroma and epithelial markers indicated in/ *tissue_segmentation.json*.	
	tisseg/overlays	overlays	Overlays of tissue segmentation masks onto composite images.	Figure 1d
	tisseg	*tissue_area.csv	Area of tissue compartments per image (pixels).	

* represents the unique identifiers of the run, such as the panel name, release or typing parameters specified in TYPEx configs.

Table 7. Visualisation of TYPEx outputs.

Outputs	Path	Output file	Description	Figure
Summary figures of marker intensities	**summary/*/intensity_plots**	cell_types_pie_chart.pdf	Pie chart of the proportion of identified cell subtypes in the analysed dataset.	/
		intensity_heatmap.pdf	Heatmap of normalised mean intensities for the cell subtype-specific markers across the identified cell subtypes in the analysed dataset.	Figure 4b
		median_intensities_per_celltype.log10.pdf	Collection of intensity distribution plots for cell subtype-defining markers between cell subtypes	Figure 2a
		median_intensities_per_positive_type.log10.pdf	Collection of intensity distribution plots between positive and negative cells for a given marker, visualised separately for each cell subtype and its confidence group.	Figure 2b

	positivity_maps.pdf	Collection of heatmaps for each cell subtype with the marker intensities and positivity calls for each subpopulation, illustrating all the unique combination of positive markers per cell subtype	Figure 4c

* represents the unique identifiers of the run, such as the panel name, release or typing parameters specified in TYPEx configs.

Table 8. Summary plots output from TYPEx.

Path	Output file	Description
summary/*/tables	cell_objects_*.txt	Cell objects with assigned cell subtype, corresponding tissue compartment, combination of positive markers
	cell_density_{panel}.txt	Cell density table summarises the cell density of each cell subtype per image. The cell densities are calculated as cell count normalised by tissue area.
	summary_*.cell_stats.txt	Table with the cell densities of each cell subpopulation per cell subtype and image with the combination of defining markers, where the cell densities are calculated per total tissue area
	categs_summary_*.cell_stats	Table with the cell densities of each cell subpopulation per cell subtype and image with the combination of markers that defines them; where the cell densities are calculated per tissue compartment.
	*.positivity.txt	Positivity table summarises the cell densities of single positive markers per image.
	*.DP.txt	Frequency summary for combinations of positive CD3, CD4 and CD8 in the dataset.
	*.unassigned.txt	Frequency summary for unassigned and ambiguous calls in the dataset.
	phenotypes.*.txt	Phenotype table shows every combination of positive markers in the dataset with the assigned cell subtype and confidence group.

Table 9. Summary tables output from TYPEx.

Tutorial 3: Spatial analysis

NOTE: When used as an independent module, Spatial-PHLEX is agnostic of all upstream processing used to derive cell identities and spatial coordinates. The user should ensure that the upstream steps used to derive the input file of cell objects should account for modality-specific processing requirements, such as removing noise, optical artefacts, performing cell segmentation and cell phenotyping.

Input data

The input data format for Spatial-PHLEX is a .csv file with four columns:

Image_ID	***X***	***Y***	***Phenotype***

!CRITICAL STEP: Column names can be arbitrary as long as they are specified in Spatial-PHLEX with the following command line flags (or in the Spatial-PHLEX nextflow.config file). For example:

```
--image_id_col
--x_coord_col
--y_coord_col
--phenotyping_column
```

Each row corresponds to a single cell in a given image. Multiple images can be represented in a single input file. See the example run command below for exact usage.

Step 1: Choose the workflow for Spatial-PHLEX to execute

Timing: 0-1 minute

Depending on the analysis the user wishes to perform on their sample, the user should specify: ‘spatial_clustering’, ‘clustered_barrier’ (or ‘default’), or ‘barrier’ in the run command. See **Experimental design** for descriptions of these workflows.

Step 2: Specify other Spatial-PHLEX input parameters

Timing: 0-5 minutes

A full description of all deep-imcyto parameters is given on the deep-imcyto documentation. Critical parameters that should be set by the user in all workflows are as follows in **Table 10**:

parameter	**description**
objects	Path to the dataframe of cell objects in csv format.
objects_delimiter	Separator for objects file. E.g. ‘,’ or ‘\t’
image_id_col	Column in objects dataframe with sample/image id for a given cell.
x_coord_col	Column header of x coordinate of cell position in image.
y_coord_col	Column header of y coordinate of cell position in image.
workflow_name	‘spatial_clustering’, ‘barrier’, ‘clustered_barrier’ (default)
release	A useful identifier for the results release.
outdir	Base output directory.

Table 10. Spatial-PHLEX input parameters which are required to be set by the user.

Workflow-specific parameters are given below in **Table 11**.

parameter	**description**	**Workflows**
eps	DBSCAN epsilon parameter. The maximum distance between two samples for one to be considered as in the neighbourhood of the other. [12] Default: 25, provides reasonable results for IMC, but may need adjustment for other imaging modalities with different pixel resolutions.	spatial_clustering, clustered_barrier
phenotyping_column	Column heading in the input data pertaining to cell phenotype information to be used for spatial clustering calculations.	spatial_clustering, clustered_barrier

phenotype_to_cluster	Which cell phenotype in the phenotyping_column to perform spatial clustering for. Select 'all' (default) to cluster all cell types in the same run.	spatial_clustering, clustered_barrier
barrier_target_cell_type	The target cell type to compute barrier scores for. E.g. Epithelial cells/	barrier, clustered_barrier
barrier_source_cell_type	The source cell type to compute barrier scores for. E.g. CD8 T cells	barrier, clustered_barrier
barrier_cell_type	The type of cell forming the barrier in the barrier scoring calculation. E.g. aSMA+ cells	barrier, clustered_barrier
barrier_phenotyping_column	Column heading in the input data pertaining to cell phenotype information to be used for barrier calculations (this can be distinct from that used for spatial clustering) (e.g. 'cellType', 'phenotype' etc) .	barrier, clustered_barrier
n_neighbours	Number of nearest neighbours for constructing the cell graph	barrier, clustered_barrier
radius	Radius cutoff for constructing spatial neighbours graph.	barrier, clustered_barrier
graph_type	Connectivity type for cell spatial graph construction (nearest_neighbour, spatial_neighbours)	barrier, clustered_barrier

Table 11. **Spatial-PHLEX parameters that the user should set for the specified analysis workflows.**

Step 3: Prepare the Spatial-PHLEX shell script

As for other modules, prepare the Spatial-PHLEX shell script from a text editor.

```
#!/bin/bash

export NXF_SINGULARITY_CACHEDIR='.singularity'

nextflow run ./main.nf \
```

```
    --workflow_name 'clustered_barrier' \
    --objects "$PWD/data/cell_objects_revision_p1.txt"\
    --objects_delimiter "\t" \
    --image_id_col "imagename"\
    --phenotyping_column 'majorType'\
    --phenotype_to_cluster 'Epithelial cells'\
    --x_coord_col "centerX"\
    --y_coord_col "centerY"\
    --barrier_phenotyping_column "majorType" \
    --barrier_source_cell_type "CD8 T cells"\
    --barrier_target_cell_type "Epithelial cells"\
    --barrier_cell_type "aSMA+ cells"\
    --n_neighbours 10\
    --outdir "../results" \
    --release 'PHLEX_testing' \
    --singularity_bind_path '/camp,/nemo'\
    --plot_palette "$PWD/assets/PHLEX_test_palette.json" \
    -w "$PWD/work"
```

Step 4: Run Spatial-PHLEX

Launch the deep-imcyto module by running the script from the command line:

```
$ ./run_Spatial-PHLEX.sh
```

Spatial-PHLEX will begin processing the input data and spatial clustering and barrier scoring outputs will be generated.

Troubleshooting of Spatial-PHLEX

Troubleshooting advice for Spatial-PHLEX can be found at https://tracerx-phlex.readthedocs.io/en/main/spatialPHLEX.html#troubleshooting.

- Spatial clusters do not represent the underlying data
    - DBSCAN clustering may not be being performed at the appropriate length scale for the coordinate data. Increase or decrease the value of the eps parameter until

cluster assignments more closely represent the spatial distribution of the underlying cells.

- Cluster boundary is too jagged / doesn't follow the shape of the cells in the cluster
    - The alpha shape parameter used to define the cell cluster boundary may not be appropriate. Increase or decrease the Spatial-PHLEX alphashape parameter until the boundary until results accord accurately with cluster boundary.

Timing for Spatial-PHLEX

The overall timing for Spatial-PHLEX depends on the workflow selected and the number of cells of interest in the images being processed. For example, barrier scoring processing times scale linearly with the number of cells of the `source_cell_type` present in the image. **Table 12** provides measured run times for Spatial-PHLEX's core workflows on the PHLEX test dataset.

	Total execution time (minutes)			
	Mean	Median	Min	Max
Spatial clustering	0.8	0.8	0.2	1.3
Barrier scoring	25.5	8.9	2.8	60

Table 12. Run times for the major components of Spatial-PHLEX applied to the PHLEX test dataset, performing spatial clustering for epithelial cells and measuring the barrier presented by αSMA$^+$ cells to CD8 T cells and epithelial cell clusters.

Anticipated results for Spatial-PHLEX

When run Spatial-PHLEX should produce the following outputs:

1. Cell cluster assignment dataframes for each unique cell type and image in the input file, including measurements of all cells to the boundary of their nearest cluster (.csv format)
2. Plots visualising the obtained spatial clusters for each cell type for the given clustering parameters (.png)

3. Binary masks of the alpha-shape boundaries of the observed clusters and colocalisation scoring
4. Intracluster density results dataframes for all cell types (.csv format) within the spatial clusters of the given cell type
5. Image-level plots of [4] (.pdf)
6. Barrier scoring results dataframes with image-level results of all barrier metrics for the given trio of cell types (.csv, see **Box 1** for a description of all metrics)
7. Single cell shortest path measurements of the source to the target cell type from which [6] are derived.

Tutorial 4: Designing an MCCS workflow for deep-imcyto's *CellProfiler* mode

Rationale for MCCS

Deep-imcyo's *simple* cell segmentation strategy which dilates nuclear masks is a useful approximation for many cells but such methods may fail to capture diverse cell morphologies or perform poorly in cell crowding areas, as well as cells that lack in-plane nuclei (**Fig 5a**). Furthermore, in alternative cell segmentation approaches where multiple channel images are combined into a reduced number of channels may result in loss of definition of cell boundaries, particularly in areas of closely packed cells. For example, using pixel classification in softwares such as Ilastik [11], a user is often encouraged to provide as input a summed or stacked image of nuclear, cytoplasmic and membrane markers from which a single channel grayscale probability map for cell area is defined, prior to instance-level segmentation. Applying watershed-like segmentation techniques, such as the propagation algorithm in CellProfiler [13], reliant on the relative value and gradients of local pixel intensity, to such single channel cell maps, may lead to artefacts arising from the production of local extrema inside neighbouring cells defined by different cell lineage markers (**Fig. 5b**).

By performing segmentation on held out cell lineage markers without first combining them, MCCS retains the steep intensity gradients between edge pixels and surrounding background pixels which can be lost when projecting channels. This reduces the likelihood of over-propagation into neighbouring cells compared to both pixel dilation and approaches relying on propagation onto all- or larger groupings of- cell types at once, which is particularly important in IMC images which

are subject to cell crowding. By combining individual cell lineage-derived cell masks, only the pixels with consensus between these masks are retained in the same cell object.

MCCS also includes an additional step to capture cells without an in-plane nucleus, as is observed for alpha-smooth muscle actin (αSMA^{+}) fibroblast populations. These populations would be missed entirely from cell segmentation in many methods relying solely on propagation outwards from nuclei onto cell boundaries (**Fig. 5**).

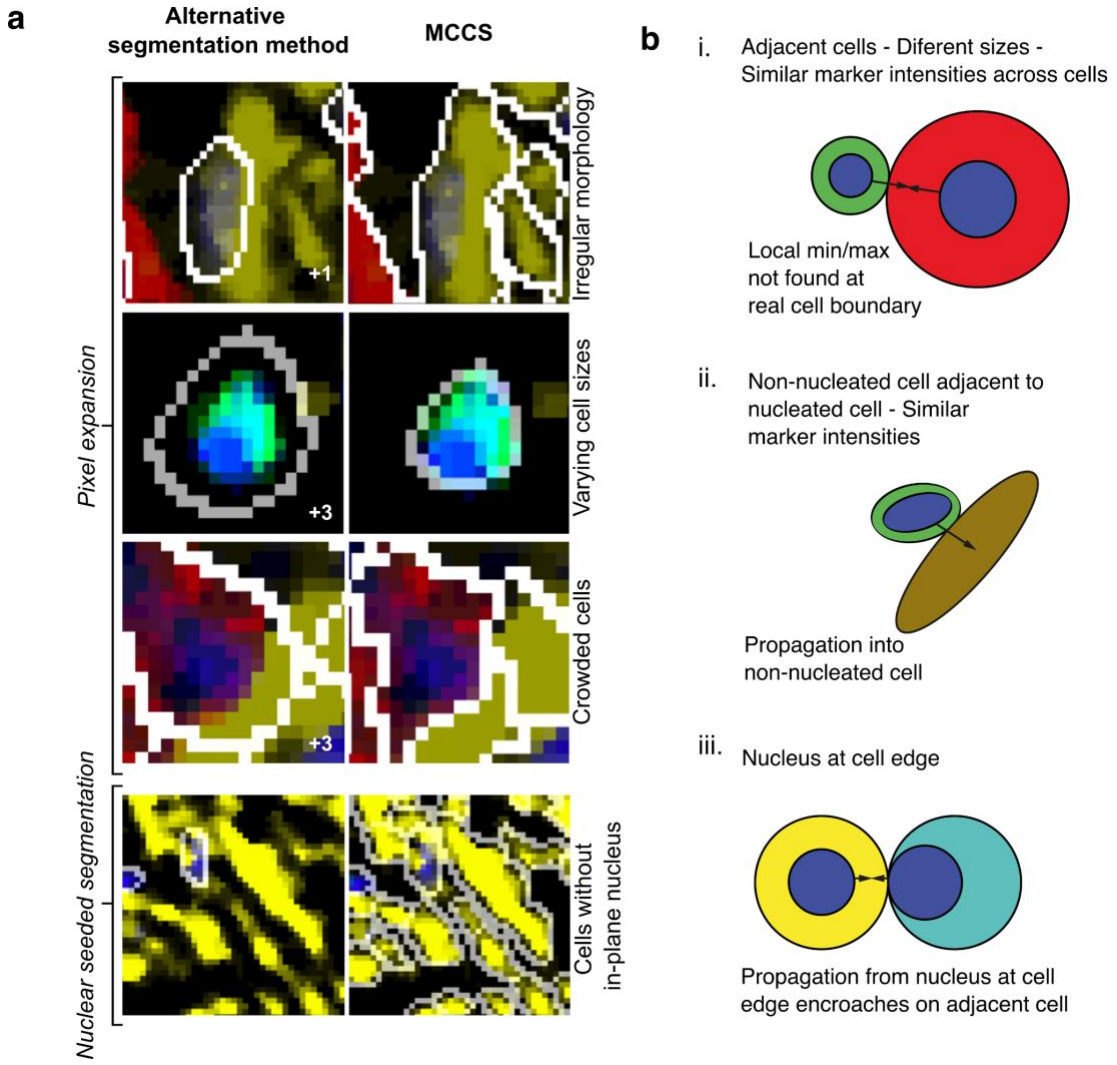


Figure 5: Multiplexed Consensus Cell Segmentation (MCCS) Rationale.

a, The rationale behind MCCS and examples of improved performance vs other classical cell segmentation methods. Numeric values in panels indicate the degree of expansion in pixels on a given nucleus during whole cell segmentation. LUAD, lung adenocarcinoma; LUSC, lung

squamous cell carcinoma; H&E, hematoxylin and eosin. **b,** Schematic representations of scenarios in which intensity-based propagation from cell nuclei onto cell cytoplasmic/membranous areas could result in inaccurate determination of cell boundaries when different cell lineage markers are used simultaneously (i.e. combined via an intensity projection into a single grayscale channel) and when cells are crowded. Different colours represent different cell lineage markers.

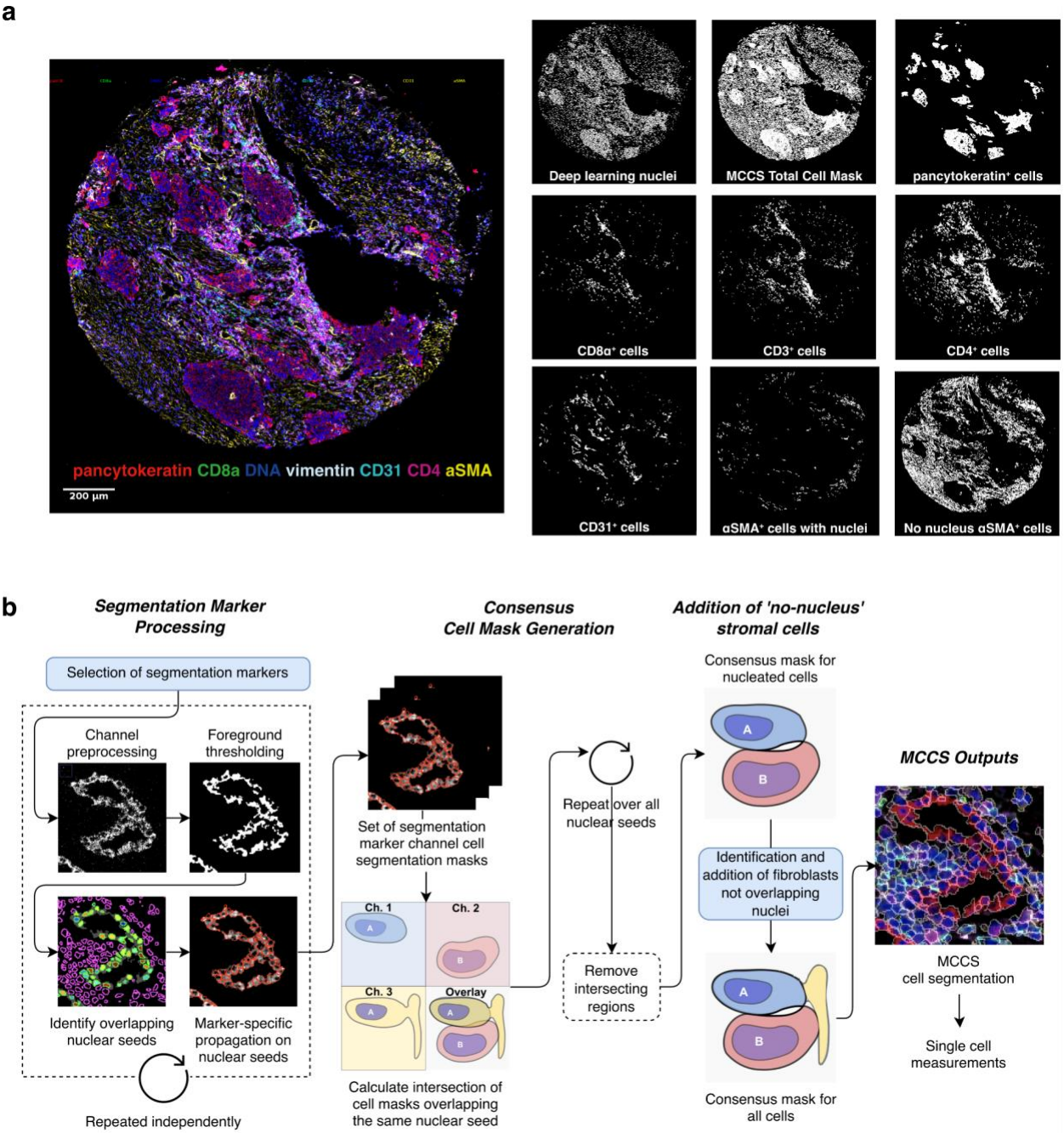


Figure 6: Multiplexed Consensus Cell Segmentation (MCCS) Workflow.

a, T cells & Stroma antibody panel composite IMC image of a lung tumour core, alongside binary masks for deep learning nuclei, the final MCCS total cell segmentation, and individual segmentation marker cell masks generated prior to combination into a final MCCS total cell mask. The extent of αSMA$^+$ cell content which does not overlap with in-plane deep learning nuclei in this core is evident from the bottom centre/right panels. All shown cells have a nucleus, apart from the 'No nucleus αSMA$^+$ cells' mask. **b,** The MCCS workflow, designed and executed in CellProfiler in the TRACERx 100 study.

Required inputs for MCCS in deep-imcyto CellProfiler mode

The required inputs for MCCS in deep-imcyto are:

- A **full_stack** of .tiff images - all channel images for which intensity measurements are to be made, with no preprocessing yet applied, generated by the IMCTools process in deep-imcyto

- An **mccs_stack** of .tiff images - a subset of the above images, defined by the user as segmentation marker images (see **MCCS Workflow Design: Segmentation marker selection** below)

- Instance-level **nuclei** segmentation .tiff images generated by the nuclear segmentation process in deep_imcyto

- Three user-configured CellProfiler .cppipe files at the filepaths specified by --full_stack_cppipe, --mccs_stack_cppipe, and --segmentation_cppipe in the deep-imcyto shell script (see **Tutorial 1** in the main manuscript). Details of how to design these pipeline files are included below. Examples are available in the assets/cppipes directory of the deep-imcyto repository (https://github.com/FrancisCrickInstitute/deep-imcyto).

- Optional: A spillover compensation matrix .tiff file, suitable for CellProfiler input. The spillover matrix is a float image with dimensions n*n (n=number of colour channels). The diagonal is usually 1 and the off-diagonal values indicate what fraction of the source channel signal is detected in other channels. R scripts used to generate such a spillover matrix for the TRACERx cohort, which used the Chevier *et al*. [14] method with minor adaptations made to the code, can be found at https://github.com/FrancisCrickInstitute/TRACERxIMCSpilllover.

- Optional: Additional CellProfiler plugin files not shipped with CellProfiler v3.1.9.

Outputs for MCCS

Expected outputs for MCCS are user-defined in the --segmentation_cppipe pipeline file in deep-imcyto, but should minimally include:

- A .csv file containing single cell level expression and/or morphometric data (cells.csv in the example MCCS implementation distributed with deep-imcyto)

- The total cell mask generated by MCCS

- Optional: .csv file containing cell neighbourhood information

Software requirements

CellProfiler [15] is a free open-source software for measuring and analysing cell images. To execute an MCCS implementation in CellPorfiler mode of deep-imcyto, a user is required to provide three user-configured CellProfiler .cppipe files at the filepaths specified by --full_stack_cppipe, --mccs_stack_cppipe, and --segmentation_cppipe.

The Docker/Singularity container which deep-imcyto uses to run MCCS runs CellProfiler v3.1.9. Accordingly, user-developed MCCS procedures should be developed using CellProfiler v3.1.9 in order to be compatible with deep-imcyto out-of-the-box.

To run an MCCS workflow developed in CellProfiler 4.x.x, the user needs to alter the deep-imcyto config to specify a different CellProfiler container which may be downloaded and built by Nextflow. Details of how to do this are made available in our online documentation at https://tracerx-phlex.readthedocs.io/en/main/deep-imcyto.html#guide-multiplexed-consensus-cell-segmentation-mccs.

To perform MCCS for any multiplexed imaging experiment, a template CellProfiler pipeline is distributed as part of PHLEX, which can be adapted to a user's own antibody panel.

Additional CellProfiler plugin files which users may find useful for building a CellProfiler MCCS procedure are bundled within the deep-imcyto repository which, if included in any of the three cppipe files, should be specified using the --plugins flag. Additional plugin files not included in the deep-imcyto release can also be deposited in this folder.

MCCS Workflow Design

A simplified overview of the key steps to implement in an MCCS workflow is included below. Steps are grouped by the input cppipe file and relevant CellProfiler modules suggested in brackets, with specific parameter values used in this study included in italics. This information is also available at https://tracerx-phlex.readthedocs.io/en/main/deep-imcyto.html#guide-multiplexed-consensus-cell-segmentation-mccs.

full_stack_cppipe

In this step, minimal preprocessing is applied to all channel images output in the imctools/full_stack/ folder.

Inputs

Each individual channel contained in the imctools/full_stack/*.tiff files should be named as inputs in the full_stack_cppipe pipeline file (**CP module: NamesAndGroups**), as should the spillover compensation .tiff if spillover compensation is being applied.

Full stack preprocessing:

If spillover compensation is being applied, this should be applied first, specifically to an ordered stack of the channels which need to be compensated (**CP modules: GrayToColor, CorrectSpilloverApply** - Spillover correction method: *NonNegativeLeastSquares.* Note: the **CorrectSpilloverApply** plugin, which is not available as a default CellProfiler v3.1.9 module, is distributed in the deep-imcyto assets.).

We then suggest removing single hot pixels in IMC images (**CP module: SmoothMultiChannel** - Neighborhood filter size *3*, Hot pixel threshold *50.0*) before rescaling all channels to between 0 and 1 - required by CellProfiler - by division by a large number which exceeds the maximum pixel intensity across any IMC channel in all images (**CP module: RescaleIntensity** - Rescaling method: *Divide each image by the same value*, Divisor value: *100000*) and saving of images in 32-bit format (**CP module: SaveImages**).

mccs_stack_cppipe

In this step, preprocessing is applied to only a subset of the full_stack images output by the IMCTools step, specifically those channels chosen as 'segmentation markers'.

Segmentation marker selection: First, a set of cytoplasmic or membrane markers is selected to capture as many cell lineages as possible in images. This should be adapted for different antibody panels and be chosen to allow for identification of different cell subsets at the segmentation stage, e.g. selecting all of CD8, CD4, and CD3 to identify distinct T cell populations. Markers with high signal-to-noise ratio (SNR) with low prevalence of artefacts across the imaged cohort should be prioritised. Markers selected in this study are summarised in **Supplementary Table S4**.

Inputs

All imctools/full_stack/ tiff files which are subject to spillover compensation should also be named as inputs in the mccs_stack_cppipe pipeline file (**CP module: NamesAndGroups**), as should the spillover compensation tiff if spillover compensation is being applied.

Segmentation marker channel preprocessing

Minimal preprocessing is first applied to all markers channels, including spillover compensation (**CP modules: GrayToColor, CorrectSpilloverApply** - Spillover correction method: *NonNegativeLeastSquares*) and hot pixel removal (**CP module: SmoothMultiChannel** - Neighborhood filter size *3*, Hot pixel threshold *50.0*) as already described.

Following this, only the subset of segmentation markers is carried forward to subsequent steps (**CP module: ColorToGray**). Median filtering is applied to these segmentation marker channels (**CP module: MedianFilter** - Window: *3*) to smooth noise to support the subsequent intensity propagation-based segmentation approach in the segmentation_cppipe. Images are then scaled across the intensity range (**CP module: RescaleIntensity** - Rescaling method: *Stretch each image to use the full intensity range*) and saved in 32 bit tiff format (**CP module: SaveImages**). We suggest naming output files in the form *preprocessed_X.tiff* where X is the segmentation marker of interest.

segmentation_cppipe

In this step, preprocessed segmentation markers are combined with the nuclear objects output from the nuclear_segmentation process in deep_imcyto to generate a total cell mask using the hybrid MCCS approach. Single-cell level metrics are then measured and produced as output alongside a total cell mask image.

Inputs

channel_preprocess/full_stack/ tiff files and channel_preprocess/mccs_stack/ tiff files should be named as inputs in the segmentation_cppipe pipeline file (**CP module: NamesAndGroups**), as should the nuclear mask image file generated by nuclear_segmentation (named with suffix _nuclear_mask.tiff by default in deep-imcyto).

Creation of a cell mask for each individual segmentation marker

The following steps are then applied to each preprocessed segmentation marker image *in turn* (i.e. independently of one another) (**Fig. 5c**, left). First, global Otsu thresholding is applied to identify areas of cell foreground based on that single cell marker (**CP module: Threshold -** Thresholding strategy: *Global*, Thresholding method: *Otsu,* Two-class or three-class thresholding: *Three classes*). Other thresholding parameters are set on a marker-by-marker basis by visual inspection across a range of representative images, which can be facilitated by CellProfiler Test

mode and GUI. This step is discussed further in **Designing an MCCS workflow: Considerations and caveats** below.

Next, the extent of overlap of thresholded cell foreground with the nuclear objects predicted by deep-imcyto is quantified (**CP module: MeasureObjectOverlap**). Specifically, nuclei for which cell foreground overlaps with at least 50% of the edge pixels of the nucleus are retained and carried forward to be used as seeds for propagation-based cell segmentation (**CP module: MaskObjects**). By requiring sufficient overlap between nuclei and cell foreground, nuclei at the edge of neighbouring cells are not inadvertently propagated out on into a neighbouring cell (**Fig. 5a**). This can be particularly important when dealing with cells with nuclei which are not centred in cell bodies (**Fig. 5a**). By requiring overlap with nuclear edge pixels only, the method is not biased against retention of large nuclei.

Retained nuclei are then used as seeds for propagation onto the relevant minimally preprocessed segmentation marker image to yield a single cell segmentation mask for that individual cell segmentation marker (**CP module: IdentifySecondaryObjects**). An individual cell in a segmentation mask can only overlap a single nucleus. Examples of segmentation-marker-specific cell masks are shown in **Fig. 5b**.

Note: The comprehensive set of CP modules required to undertake these steps can be found in the template segmentation.cppipe CellProfiler pipeline file distributed with PHLEX.

Consensus Cell Mask generation

Next, having repeated the above series of steps for each segmentation marker independently, the set of single segmentation marker cell masks is combined (**Fig. 5c**, middle). First, nuclei not retained as seeds in any cell segmentation mask are dilated on by +1 pixel expansion to reflect the expected larger area of a cell than its contained nucleus for cells for which no appropriate cell segmentation marker is available (**CP module: DilateObjects**). A +1 pixel expansion was determined to result in less signal bleed between neighbouring cells and improved signal capture for markers compared to a +3 expansion in our previous work [9]. Only areas of dilated nuclei not overlapping individual segmentation marker cell masks are retained (**CP module: MaskObjects**).

Then, a series of serial maskings is performed such that any pixels retained in the same cell object must be found in the same set of segmentation marker masks (**CP module: MaskObjects**). This step ensures that only pixels with consensus between different marker channels are retained in the same single cell object, and that individual pixels which could derive from different cell types

at cell boundaries are removed from analysis to avoid inadvertently assigning a cell membrane marker to the wrong one of two tightly packed neighbouring cells. This whole step yields a total nucleated cells segmentation mask.

Note: The comprehensive set of CP modules required to undertake these steps can be found in the template segmentation.cppipe CellProfiler pipeline file distributed with PHLEX.

Optional - Addition of non-nucleated cell populations

Finally, an additional primary object identification step can be applied directly to any single channel observed to stain cells which lack coincident nuclei in the imaging plane (**CP module: IdentifyPrimaryObjects**). For example, in the TRACERx study, αSMA^{+} cells were frequently observed without an in-plane nucleus. Thresholding (*global Otsu*) and cell size parameters (*min 20 pixels, max 1000 pixels*) are optimised by visual inspection to identify these additional 'non-nucleated' cell populations, and then identified objects combined with the total nucleated cell segmentation mask, removing areas of overlap and non-nucleated objects with >33% overlap with the nucleated cell mask (**CP module: MaskObjects**).

Note: The comprehensive set of CP modules required to undertake these steps can be found in the template segmentation.cppipe CellProfiler pipeline file distributed with PHLEX.

Finally, minimum (*4 pixels*) and maximum (*1000 pixels*) cell area filters are applied to yield a final total cell segmentation mask (**CP module: FilterObjects**) (**Fig. 5b,c)**.

Extraction of single cell-level metrics

This final total cell segmentation mask is then used to delineate the boundaries of individual cells and extract single cell-level expression data and morphometric features (**CP modules: MeasureIntensityMultichannel, MeasureObjectSizeShape**). The single cell expression data measured can be fed into TYPEx for identification of cellular phenotypes and cell marker positivity status. If desired, additional (CellProfiler) modules can also be implemented on the final total cell mask to derive information about cell-cell neighbour relationships (**CP module: MeasureObjectNeighbor**).

Running an MCCS workflow with deep-imcyto in CellProfiler mode

Having designed the three MCCS workflow .cppipe files described and confirmed input and software requirements are fulfilled, MCCS can be invoked in the deep-imcyto framework by preparing a shell script as described in **Tutorial 1: Step 4: Prepare the deep-imcyto shell script** and running the pipeline as described in **Tutorial 1: Step 5: Run deep-imcyto**.

Example timings for an MCCS workflow designed for the TRACERx 100 cohort can be found in the section **Tutorial 1: Timing for deep-imcyto**.

Designing an MCCS workflow: Considerations and caveats

Using other software to design and implement an MCCS workflow

While we developed a Multiplexed Consensus Cell Segmentation (MCCS) procedure in CellProfiler [15] in the TRACERx study, in principle, the MCCS approach can be executed outside of deep-imcyto in any appropriate software which permits thresholding and propagation-based object identification on images, and for any multiplexed imaging technology, including immunofluorescence-based approaches.

A trio of CellProfiler pipeline files for full_stack_preprocessing, mccs_stack_preprocessing, and segmentation are distributed as part of PHLEX. These workflows can be adapted to a user's own antibody panel and provide a template for other software implementations.

However it should be stressed that the deep-imcyto nuclear segmentation framework is designed to be used on IMC data only and therefore it would not be appropriate to invoke a custom MCCS workflow designed for a different multiplexed imaging modality using the deep-imcyto framework. In such cases, alternative methods outside of deep-imcyto should be used to segment seeding nuclei for data acquired from imaging modalities other than IMC.

Selecting the large number for rescaling in full stack preprocessing

As mentioned in the **Tutorial 4: Designing an MCCS workflow in CellProfiler**, in order to save all .tiff files in the full_stack_preprocessing .cppipe pipeline, all channel images must be rescaled between 0 and 1 by division by a large number which exceeds the maximum pixel intensity across any IMC channel in all images. A divisor value of 100000 was found to be sufficient in the

TRACERx 100 study for both antibody panels. Analysis of raw pixel intensities in a subset of images may help to guide user selection of an appropriately large value. Alternatively, a user might consider running the deep-imcyto QC workflow to generate raw IMC images before running a script to determine such a maximum value automatically.

Selection of segmentation markers

The selection of cell lineage markers with appropriately high SNR and specificity, and low prevalence of artefacts, should be undertaken over an appropriately large representative set of images and therefore can be time-consuming. Prior biological knowledge of which markers on the panel are likely to represent cells (as opposed to acellular structures such as extracellular matrix) and could putatively represent the greatest number of distinct cell lineages is required.

Optimisation of parameters by visual inspection

In designing an MCCS procedure, a user is encouraged to use visual assessment to optimise thresholding parameters to identify foreground for each segmentation marker across a representative range of cores (**Fig. 5c, left**). Similarly, additional visual inspection should be undertaken on cell masks subsequently derived from individual segmentation markers, using marker channels with masks overlaid, to optimise secondary object identification parameters (**Fig. 5c, left**).

The validation of thresholding and segmentation parameters for each marker, can be time-consuming and requires subjective decision-making, for example, in assessing specificity, in determining cell foreground compared to background and in deciphering cell boundaries.

The authors acknowledge such caveats to the MCCS approach, and suggest mitigating these difficulties through appropriate selection of softwares with GUIs (such as CellProfiler [15] or FIJI ImageJ [16]) which allow interactive testing of different parameter values and concurrent visual inspection, as well as use of more than one user to minimise individual subjective bias. Additionally, whilst we have chosen to employ visual inspection using the CellProfiler GUI in this study, we present the MCCS procedure as a series of steps which can be invoked using different softwares as deemed more user-friendly by an individual user. For example, an alternative approach could be to use a pixel classification software such as Ilastik [11] which reports error metrics to provide a more quantitative confidence measure during semantic thresholding of cell foreground.

Despite these considerations, it should be noted that the design of an MCCS pipeline, whilst time-consuming, is proposed to replace the need for manual labelling of cell boundaries for machine-learning based segmentation model training, which is difficult in IMC experiments due to noise, low image resolution and the question of how an appropriate image to label should be constructed, and itself time-consuming. Few large-scale labelled IMC datasets have been published, and those that have contain biases towards particular tissue types and histologies, potentially limiting their application to a new user's dataset.

Nuclear overlap criterion

While segmentation markers are expected to localise to cell cytoplasm and membranes and would therefore not be expected to coincide with nuclear masks, given the low resolution of IMC, we frequently saw overlap of these markers with pixels at the edge of segmented nuclei. We note that previous IMC studies have also relied on nuclear overlap criteria to segment cells by marker, such as in the sequential segmentation pipeline from van Maldegem *et al*. [9]. Here, we specifically enforce that overlap occurs with nuclear edge pixels recognising that we would not expect cytoplasmic/membrane indicators to overlap with central nuclear pixels.

The 50% overlap criterion between nuclei edges and thresholded segmentation marker foreground was selected to trade off two competing demands and was determined to be appropriate for the chosen indicator channels by visual inspection:

1. **The assumption that a nucleus would likely have greater overlap between nuclear edge pixels and the indicator marker(s) for that cell than other indicator marker(s) not indicative of that cell type.** For example, the more stringent the nuclear overlap the less likely a nucleus could be wrongly assigned to multiple mutually exclusive indicator markers in crowded cell regions (e.g. panCK and CD8, or CD8 and CD4), due to, for example, an acentric nucleus encroaching on an adjacent cell (**Fig. 5a**).

2. **The observation that, even in cells characterised by an indicator marker, there is not 100% overlap with nuclear edge pixels**, in part due to the complementary nature of indicator markers and nuclei, and in part the noisy nature of IMC data.

Cell size criteria

Regarding cell size, in our study, a minimum nucleus size of 4 pixels and cell size of 9 pixels (representing a 3x3 micron area) and maximum cell size of 1000 pixels were included to represent the expected range for such features in images. Values may need to be adapted for a new user's experiment.

Parameters for identifying non-nucleated cells

The parameters used in the segmentation of non-nucleated stromal cells were optimised by visual inspection over a range of cores. Size filters were imposed on identified non-nucleated cells and non-conforming cells excluded to avoid picking up individual pixels or small artefacts - (i) a minimum cell diameter of 5 pixels and maximum of 35 pixels during primary object identification, and (ii) a minimum cell size of 20 pixels and a maximum cell size of 1000 pixels for the leftover cell areas following subsequent masking by the nucleated cells mask. Additionally, only non-nucleated αSMA^{+} cells having a maximum overlap of 33% with nucleated cell masks were retained. This overlap criterion was implemented to avoid inadvertently doubly counting cells where a small area of αSMA stain extended beyond the boundaries of a full nucleated cell compared to other segmentation marker stains. Areas of overlap between the nucleated and non-nucleated cell masks were masked out.

αSMA used in this study but can be replaced by other channel(s) for a marker(s) observed to be expressed by cell populations frequently observed without an in-plane nucleus, adjusting cell size parameters accordingly.

References

1. Schürch, C. M. *et al.* Coordinated Cellular Neighborhoods Orchestrate Antitumoral Immunity at the Colorectal Cancer Invasive Front. *Cell* **183**, 838 (2020).
2. Brbić, M. *et al.* Annotation of spatially resolved single-cell data with STELLAR. *Nat. Methods* **19**, 1411–1418 (2022).
3. Black, S., Phillips, D., Hickey, J. W. & Kennedy-Darling, J. CODEX multiplexed tissue imaging with DNA-conjugated antibodies. *Nat. Protoc.* (2021).

4. Jamal-Hanjani, M. *et al.* Tracking the Evolution of Non-Small-Cell Lung Cancer. *N. Engl. J. Med.* **376**, 2109–2121 (2017).
5. Ghorani, E. *et al.* The T cell differentiation landscape is shaped by tumour mutations in lung cancer. *Nat Cancer* **1**, 546–561 (2020).
6. Enfield, K. S. S. *et al.* Spatial Architecture of Myeloid and T Cells Orchestrates Immune Evasion and Clinical Outcome in Lung Cancer. *Cancer Discov.* (2024) doi:10.1158/2159-8290.CD-23-1380.
7. Jackson, H. W. *et al.* The single-cell pathology landscape of breast cancer. *Nature* vol. 578 615–620 Preprint at https://doi.org/10.1038/s41586-019-1876-x (2020).
8. Greenwald, N. F. *et al.* Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. *Nat. Biotechnol.* **40**, 555–565 (2022).
9. van Maldegem, F. *et al.* Characterisation of tumour microenvironment remodelling following oncogene inhibition in preclinical studies with imaging mass cytometry. *Nat. Commun.* **12**, 5906 (2021).
10. Bind Paths and Mounts — Singularity container 3.0 documentation. https://docs.sylabs.io/guides/3.0/user-guide/bind_paths_and_mounts.html.
11. Berg, S. *et al.* ilastik: interactive machine learning for (bio)image analysis. *Nat. Methods* **16**, 1226–1232 (2019).
12. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *arXiv [cs.LG]* 2825–2830 (2012).
13. McQuin, C. *et al.* CellProfiler 3.0: Next-generation image processing for biology. *PLoS Biol.* **16**, e2005970 (2018).
14. Chevrier, S. *et al.* Compensation of Signal Spillover in Suspension and Imaging Mass Cytometry. *Cell Syst* **6**, 612–620.e5 (2018).

15. Stirling, D. R. *et al.* CellProfiler 4: improvements in speed, utility and usability. *BMC Bioinformatics* **22**, 433 (2021).
16. Rueden, C. T. & Eliceiri, K. W. ImageJ for the Next Generation of Scientific Image Data. *Microsc. Microanal.* **25**, 142–143 (2019).

Acknowledgements

We are grateful for assistance from Zoe Ramsden, Mark Hill, Hanyun Zhang, Marcellus Augustine, and Febe van Maldegem. The authors acknowledge the use of BioRender.com to create illustrations in **Figure 1a**.