ContextExplorer Documentation

Release 1.0

Joel Ostblom

Mar 15, 2019

Contents:

Context-explorer is a software program that facilitates analyses of data extracted from microscope images of cells.

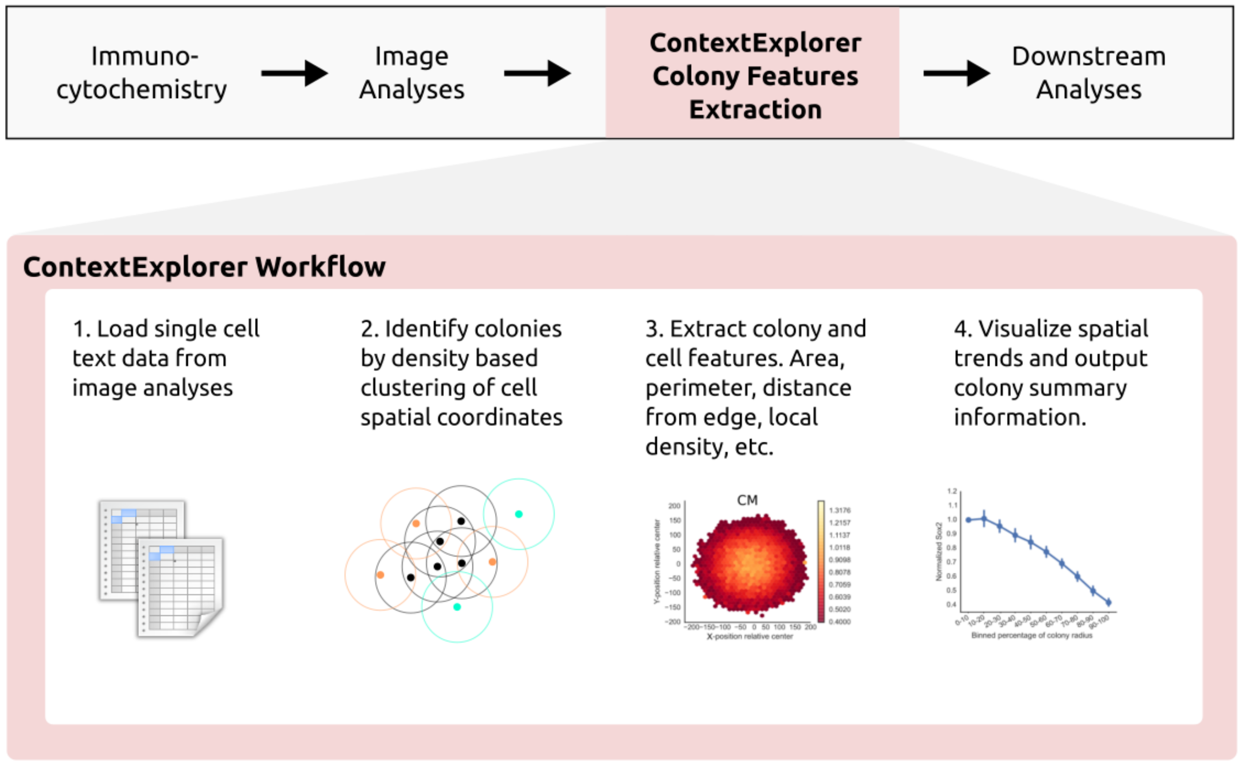
CHAPTER 1

Motivation

The analyses methods in context-explorer focuses on how populations of cells are affected by their microenvironment, including local variations in cell signalling and cell density. It is currently difficult for scientist without previous programming experience to quantify these variables, although it is relevant for many research areas. Facilitating this type of analyses can help scientist around the world to improve their understanding of cellular behavior and accelerate their research.

CHAPTER 2

Graphical overview


Immuno-cytochemistry
Image Analyses
ContextExplorer Colony Features Extraction
Downstream Analyses
ContextExplorer Workflow
1. Load single cell text data from image analyses
2. Identify colonies by density based clustering of cell spatial coordinates
3. Extract colony and cell features. Area, perimeter, distance from edge, local density, etc.
4. Visualize spatial trends and output colony summary information.
CM
X-position relative center
Y-position relative center
Normalized Sox2
Binned percentage of colony radius

CHAPTER 3

Installing Context-explorer

The fastest way to get up and running, is to install context-explorer via the `conda` package manager from the Anaconda Python distribution:

1. Download and install Anaconda Python for your platform.

2a. Open the Anaconda navigator and search for context-explorer

2b. Open a terminal or the Anaconda Prompt and type `conda install -c joelostblom context-explorer`.

CHAPTER 4

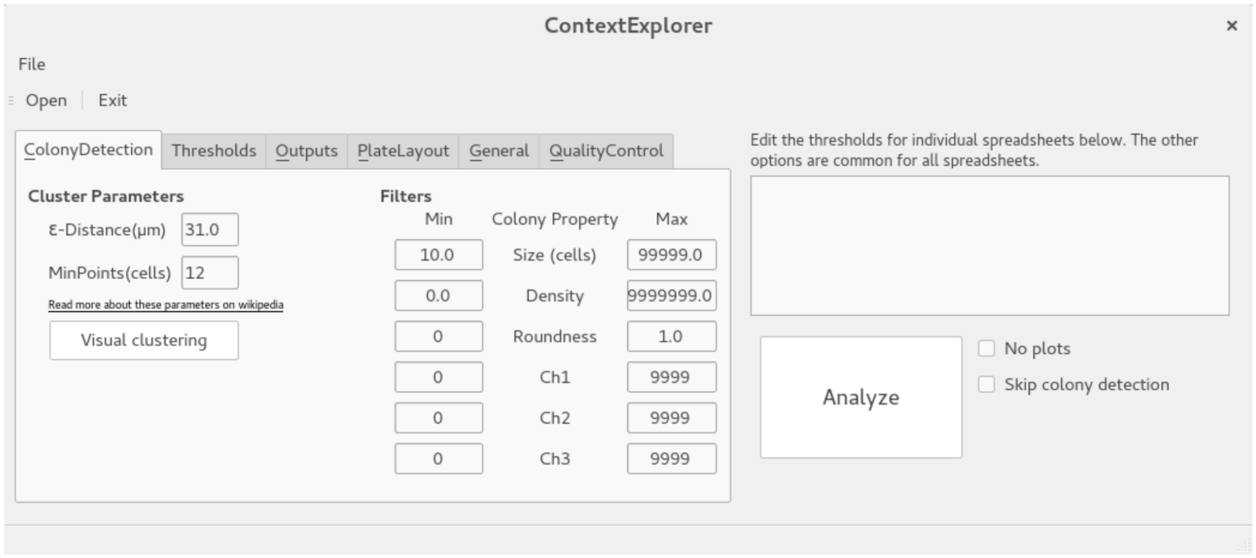
Using context-explorer

Launch the context-explorer by opening a terminal/the `Anaconda Navigator`, typing `ctexplorer`, and hitting enter. Keep the terminal window open, there is where context-explorer will output messages of what it is doing.

4.1 Input file

The minimum required input is a CSV-file containing single cell measurements including xy-coordinates and field number for each cell. Such measurements can be extracted from microscope images using software such as CellProfiler or vendor-specific alternatives. The field number represents the position of the tile/site within each well and the xy-coordinates are the position of each cell within these tile. Ideally, at least one additional measure of interest is present, e.g. fluorescent intensity or nuclear area.

4.2 Workflow overview



Cluster Parameters

ε-Distance(μm) 31.0

MinPoints(cells) 12

Filters

Min	Colony Property	Max
10.0	Size (cells)	99999.0
0.0	Density	9999999.0
0	Roundness	1.0
0	Ch1	9999
0	Ch2	9999
0	Ch3	9999

Edit the thresholds for individual spreadsheets below. The other options are common for all spreadsheets.

There is a brief video tutorial of how to use the software.

After loading the desired CSV-file, assign treatment groups via the plate layout tab. In the threshold tab, it is possible to create histograms and scatter plots of the desired measurements. One plot will be created per well. Basic column modifications (e.g., addition and deletion) can be performed via the ColumnModification tab.

In the colony identification tab, cells can be grouped into clusters based on density variations within each well. The clustering parameters can be adjusted interactively, so that the clusters represent biologically relevant entities, e.g. colonies of stem cells or micropatterned cells. This grouping is helpful to assess the variation of the cellular microenvironment on cell fate through comparisons between groups at distinct spatial locations within the clusters, .e.g. comparing the expression of certain proteins between cells growing at the colony edge vs the middle of the colony.

The data can then be saved via the SaveData tab either as single cells data or aggregated at the level of choice (e.g. average measurements per well or treatment group).

4.3 Package build instructions

These instructions are not needed if you just want to use `context-explorer`. The steps below details how to build a Python package (1-2) and upload it to https://pypi.org (3), and how to build a `conda` package (4-6) and upload it to https://www.anaconda.com/download (7).

1. Create a git tag with the version number.

   ```
   git tag <major.minor.patch>
   ```

   If there are any changes in the working tree since this tag, these needs to be stashed or `versioneer` will create a version tag that is incompatible with PyPI.

2. Build the source distribution (for `conda` to use in the next step) and at least one binary wheel (faster for `pip install`).

   ```
   python setup.py sdist bdist_wheel
   ```

3. Upload the package to PyPI.org.

```
twine upload dist/*<major.minor.path>*
```

4. A `conda` build recipe can be created from the PyPI package.

   ```
   conda skeleton pypi context-explorer
   ```

   This will create the directory `context-explorer` with the file `meta.yml`. The only manual modification that needs to be made is to replace the occurrences of *pyqt5* with *pyqt*, since this package is named differently on `PyPI` and in `conda`:

5. Build a `condo` package from the recipe.

   ```
   conda build context-explorer
   ```

6. The conda package is usually created in `~anaconda3/conda-bld/linux-64/<package_name>` and can be converted to support all platform architectures.

   ```
   conda convert --platform all <absolute/path/to/package> -o <outputdir>
   ```

7. Each platform's package needs to be uploaded separately to anaconda.org.

   ```
   anaconda upload <path/to/package>
   ```

   The initial package created with `conda build` can be uploaded automatically by setting the following.

   ```
   conda config --set anaconda_upload yes
   ```

CHAPTER 5

Indices and tables

Heading 1

- genindex
- modindex

Heading 2

- search