Documentation for qPAINT analysis

This guideline should help you through processing DNA-PAINT images to convert single-molecule localisations into protein maps and get quantitative information out (i.e., cluster properties and nearest neighbour distances distribution). This works for either 1 colour data or 2 colour data.

If you have any questions, please contact Dr. Sabrina Simoncelli (s.simoncelli@ucl.ac.uk) or Megan Joseph (megan.joseph.20@ucl.ac.uk).

Requirements:

- MATLAB version no earlier than 2014b
  - http://uk.mathworks.com/products/matlab/?refresh=true
- RegionFinder (Installed as an APP in MATLAB)
  - https://github.com/quokka79/RegionFinder
- PALMsiever (Install with instructions without DipImage)
  - https://github.com/PALMsiever/palm-siever/wiki

Code:

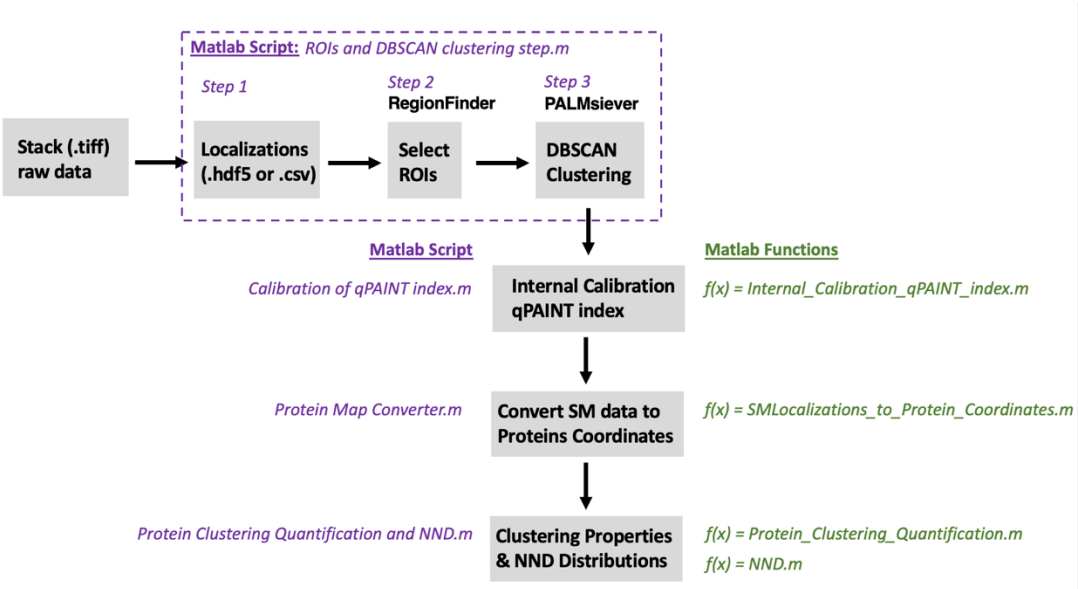

Figure 1: Analysis workflow depicting the main functions and their relation to the data processing stages.

Data analysis is performed via a combination of DBSCAN cluster analysis and self-written Matlab routines to: (i) calculate the calibration qPAINT index from the raw data (aka: Internal Calibration); (ii) convert the single-molecule localisation files into files containing the coordinates with the best estimation of the proteins positions (via qPAINT and k-means); (iii) calculate clustering properties (i.e., number of cluster, percentage of proteins in clusters,

cluster size, etc..) and nearest neighbour distances (NND) for one or two-colour data sets. To run the Matlab routines associated to these operations pleases use:

(i) Calibration of qPAINT index.m file
(ii) Protein Map Converter.m
(iii) Protein Clustering Quantification and NND.m

Several functions are required to run these routines, these functions are provided in the Functions folder. However, please leave the functions inside the same folder as the routines that you will be running. Note, that any change of user input required to run the functions can be made within the routine's codes, there is no need to modify any function.

For data preparation, i.e., running selecting regions of interest (ROIs) in different cells and running DBSCAN please use the routine: ROIs and DBSCAN clustering steps.m

Implementation

(1) Data preparation - Cropping ROIs:

- Convert HDF5 files (of ROIs or cells) to .csv files if need be by running the MATLAB code titled ROIs and DBSCAN clustering steps.m (Step 1). This code will save the: *x, y, frame, photon, sx (error in x) and sy (error in y)* columns from the HDF5 files in that order.
- Open RegionFinder in MATLAB by going to Apps and clicking on the RegionFinder App.
- Open your .csv cell file in RegionFinder by selecting XY only, image size of around 60 (you may need to play with this), 1nm/px (we already converted them), and a data delimiter of comma.
- Select the columns of your x,y, and frame data which should be x=1, y=2, channel ID = 0 and frame = 3
- Select the size of the ROI in region size e.g. 3000 is 3 micrometers by 3 micrometers
- Make sure hide first is unselected
- Then click open one file and select the file to load
- Then click new region, a square should appear, click and drag the square to where you want your ROI and double click to select the region
- Once you have finished selecting the ROIs save them by selecting export regions
- This should save a coords.txt file that you can use the cut regions of the cell in the next section of code
- Save the coords.txt file also in this folder
- Open the coords.txt file (in any text file viewer) and change the first column to be 1, change the second column to be the ROI number e.g. 1 for the first row, 2 for the second, and so on
- Run ROIs and DBSCAN clustering steps.m (Step 2) and select the coords.txt file when requested.

(2) Data preparation – Running DBSCAN:

- Open PALMsiever in MATLAB by navigating to the file where PALMsiever is saved in your computer and typing PALMsiever into the command line.
- Load your .csv file of the cell/ROI by clicking file, import, generic text file

- Go to plugins, cluster_DBSCAN to start DBSCAN clustering and input the parameters for “minpts” and “eps”, this will start the clustering algorithm, you won’t see anything until the algorithm is finished, then it will show the clustered output.
- Save the DBSCAN output to a .txt file by running the MATLAB code titled ROIs and DBSCAN clustering steps.m (Step 3). This code will save the: *frame, x, y, photons, error in x, error in y, dbscan_ID, dbscan_type* into a file with the name of the Eps and MinPts parameters in.
- Make sure to save the data in the following format: Cell1_ROI1_Ch1_DBSCAN_Ex_Py.txt (see Figure 2).

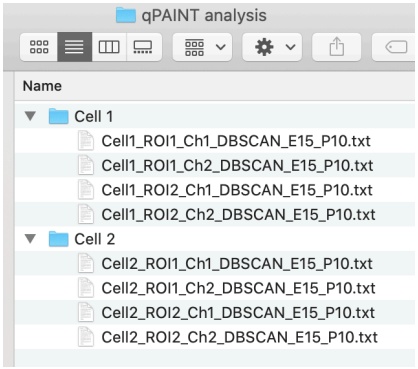

Figure 2. Setting DBSCAN files of each ROIs corresponding to each FOV (field of view) or Cell in the correct format for posterior analysis.

(3) Internal Calibration qPAINT index

Overview: Calculates qPAINT index (qi) (i.e., inverse of dark times) for all clusters with a maximum point distance of 150 nm in each ROI using the information obtained after running DBSCAN cluster analysis on the DNA-PAINT single-molecule localisation microscopy data.

In short, time stamps (frame number) of localisations within the same cluster are used to reconstruct the sequence of dark times per cluster as continuous frame times that did not contain an event. All the dark times per cluster are pooled and used to obtain a normalised cumulative histogram of the dark times which is then fitted to 1 - exp(t/tau_d). This allows to estimate the dark time, tau_d, per cluster.

- Open the MATLAB code Calibration of qPAINT index.m
- Input the correct user parameters into the code as requested (Step 1)
- This step will generate a new .csv file in each subfolder where the ROIs are stored with the “_Internal Calibration_150” addition. The output file is structured as follows:
  - ⇒ Column 1: clusterID
  - ⇒ Column 2: locs per cluster
  - ⇒ Column 3: mean frames
  - ⇒ Column 4: darktimecounts
  - ⇒ Column 5: darktime_mean
  - ⇒ Column 6: muMLE = darktime via cumulative distribution function fit
  - ⇒ Column 7: qPAINT_index for each cluster. The value is provided as 100x in 1/s

- It will also generate a merged file containing all the qPAINT indexes (Column 7) of all the files generated in each Cell or FOV subfolders (see Figure 3 with results outputs).

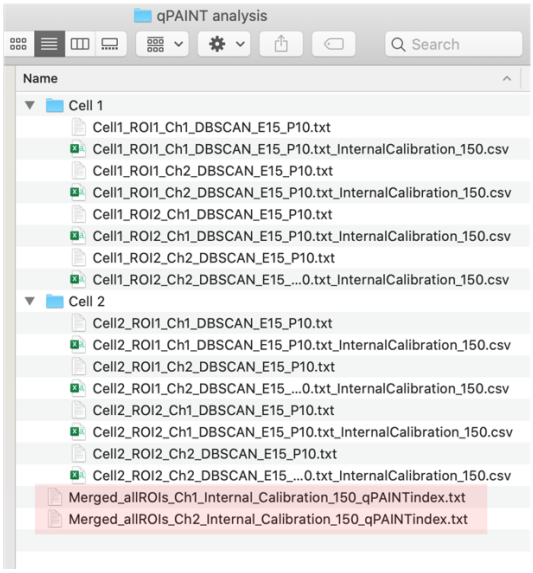

Figure 3. Output files of Internal Calibration step for each ROIs corresponding to each FOV (field of view) or Cell.

- Use the merged file:
  'Merged_allROIs_Ch1_Internal_Calibration_150_qPAINTindex.txt'
  to plot the histogram with all the qPAINT index and determine the qPAINT index corresponding to 1 docking site (i.e., 1 protein).
- Run the MATLAB code Calibration of qPAINT index.m (Step 2) and go to apps, Curve fitting as explained in detailed in the script.

(4) Convert Single-Molecule Localisations into Protein Coordinates

Overview: Generates x,y molecular coordinates from the DNA-PAINT source data after running running DBSCAN (input files are in the format of Cell1_ROI1_Ch1_DBSCAN_E15_P10.txt).

To convert SM localizations into molecular coordinate the function uses k-means clustering and the qPAINT index value corresponding to one single docking site (i.e., 1 protein). This value is obtained in previous step (3).

The output of running this code: MATLAB code Protein Map Converter.m (Step 1) are two new .csv files in each of the subfolders where the ROIs are stored with the "_Protein_Maps" and "_Protein_Info".
These files are structured as:

(1) "_Protein_Maps": Column 1 and 2: x, y coordinates of proteins in nm.
(2)"_Protein_Info":
⇒ Column 1: clusterID
⇒ Column 2: locs per cluster
⇒ Column 3: mean frames
⇒ Column 4: darktimecounts
⇒ Column 5: darktime_mean
⇒ Column 6: muMLE = darktime via cumulative distribution function fit
⇒ Column 7: qPAINT_index for each cluster. Value is provided as 100x in 1/s.

⇒ Column 8: maxClusterDistance.
⇒ Column 9: number of proteins per cluster

- Open the MATLAB code Protein Map Converter.m
- Input the correct user parameters into the code as requested, ensuring you input the qPAINT index calculated for each ChID (colour 1 and 2 if having more than single colour data) and run.
- The output files correspond to the ones highlighted in Figure 4.

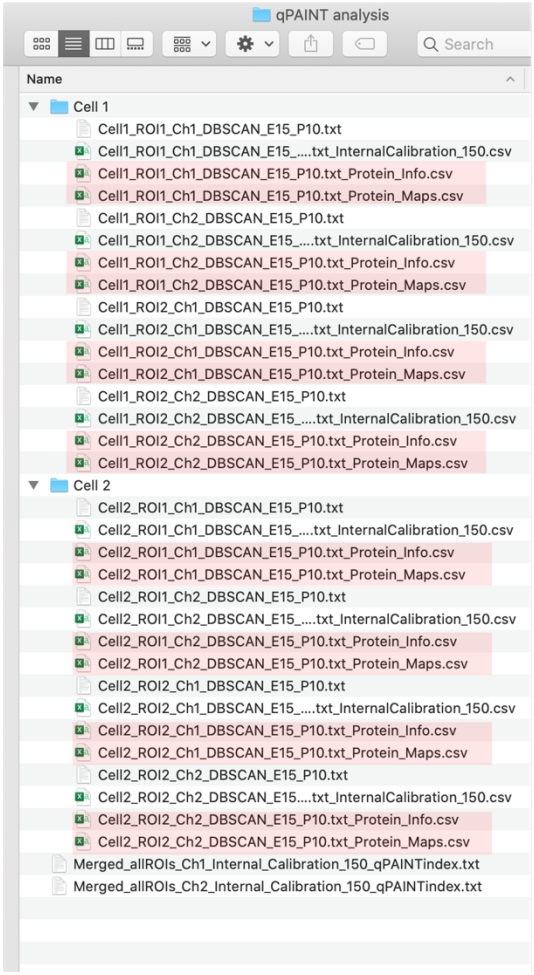

Figure 4. Output files of Protein Coordinates step for each ROIs corresponding to each FOV (field of view) or Cell.

(5) Calculate Protein Clustering Information

Overview: Calculate different parameters associated to protein cluster quantification to compare conditions (i.e. KO vs WT; activated vs non-activated, etc).

The Protein Clustering Quantification and NND.m (Step 1) calculates the following parameters:

⇒ Total Number of Proteins: 'TotalProteins.txt'
⇒ Number of Clusters (cluster defined as > 3 proteins): 'NumberofClusters.txt'
⇒ Percentage of proteins in clusters: 'PercetangeClusteredProteins.txt'
⇒ Cluster Size (max distance btw two points in the cluster): 'ClusterSize.txt'
⇒ Cluster Size in terms of number of proteins (number of proteins per

cluster): 'NumProteinsperCluster.txt'

⇒ Number of different Cluster Types (i.e. Small, Medium, Large):'ClusterTypes.txt'

In each file you will have in the 1st Column the quantity per ROI, and if applicable, in the 2nd Column the quantity per micrometer^2.

- Open the MATLAB code Protein Clustering Quantification and NND.m
- Input the correct user parameters into Step 1 as requested and run.
- The output files correspond to the ones highlighted in Figure 5.

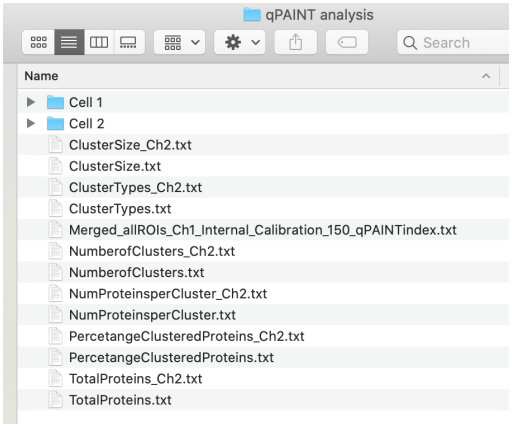

Figure 5. Output files of Protein Clustering Quantification for all the ROIs in all the subfolders corresponding to the different FOVs (field of view) or Cells.

(6) Calculate Nearest Neighbour Distance Distributions.

Overview: Nearest neighbour distances (NND) for proteins of the same identify (i.e, Ch1-Ch1; Ch2-Ch2) or different types (i.e, Ch1-Ch2) are calculated using the proteins maps obtained via qPAINT analysis and k-means clustering (using Protein Map Converter.m code, and files with the _Protein_Maps.csv extension).

To evaluate the significance of the NND distributions (i.e., Ch1-Ch1, Ch2-Ch2, Ch1-Ch2, Ch2-Ch1) this code also generates random positions for each protein type to calculate the random NND distribution for each case. The resulting histogram of the nearest neighbour distances for both experimental data sets and the randomly distributed data is then normalized using the total number of NND calculated per ROI to quantify the percentage of the populate with distances smaller than a user-defined set threshold value(s).

The code generates the following output files by merging together the information of all the ROIs in all the Cells/FOVs analysed:

Distribution of Nearest Neighbour Distances (NND):

(1) NND Ch1 to Ch1 for data and Rnd:'NND_Ch1.txt' & 'NND_Ch1_Rnd.txt'
(2) NND Ch2 to Ch2 for data and Rnd:'NND_Ch2.txt' & 'NND_Ch2_Rnd.txt'
(3) NND Ch1 to Ch2 for data and Rnd:'NND_Ch1_Ch2.txt' & 'NND_Ch1_Ch2_Rnd.txt'
(4) NND Ch2 to Ch1 for data and Rnd:'NND_Ch2_Ch1.txt' & 'NND_Ch2_Ch1_Rnd.txt'

Summary descriptors of distributions for data and random distributions:

(5) Summary of NND Ch1 to Ch1 for data and Rnd:'Summary_NND_Ch1'
(6) Summary of NND Ch2 to Ch2 for data and Rnd:'Summary_NND_Ch1'
(7) Summary of NND Ch1 to Ch2 for data and Rnd:'Summary_NND_Ch1_Ch2.txt'
(8) Summary of NND Ch2 to Ch1 for data and Rnd:'Summary_NND_Ch1_Ch2.txt'

The summary files are organised as follows; each row corresponds to one ROI:

⇒ Column 1 & 2: Average value of the NND for data and random
⇒ Column 3 & 4: Median value of the NND for data and random
⇒ Column 5 & 6: % of NND below threshold T1 (i.e., 25 nm) for data and random
⇒ Column 7 & 8: % of NND below threshold T2 (i.e., 50 nm) for data and random.

- Open the MATLAB code Protein Clustering Quantification and NND.m
- Input the correct user parameters into Step 2 as requested and run.
- The output files correspond to the ones highlighted in Figure 6.

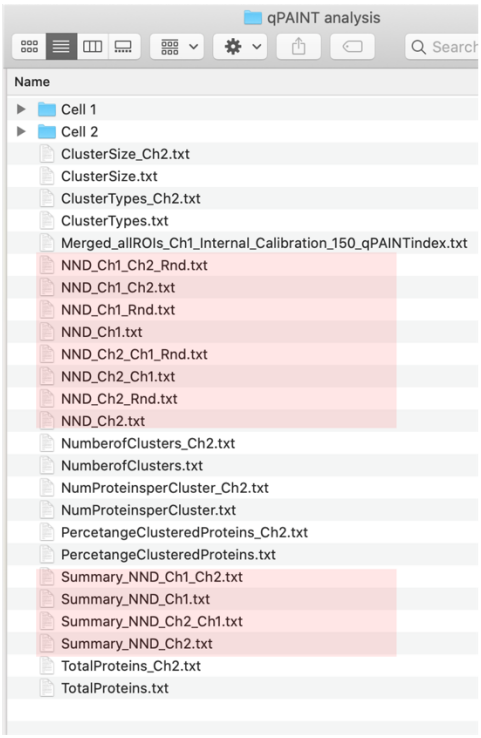

Figure 6. Output files of NND calculations for all the ROIs in all the subfolders corresponding to the different FOVs (field of view) or Cells.