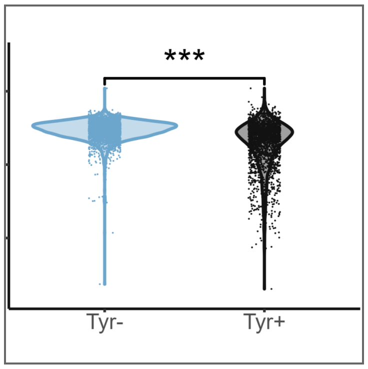


JAN 31, 2024

Neuromelanin quantification in stained brain slices of the Locus coeruleus

Csilla Novák[1], Maria P. Contreras[1], Matthias Prigge[1]

[1]Neuromodulatory Network Group, Leibniz Institute for Neurobiology, Magdeburg

ASAP Collaborative Research Network   TeamPrigge

priggelab

DOI:
dx.doi.org/10.17504/protocols.io.5qpvo32y9v4o/v1

Protocol Citation: Csilla Novák, Maria P. Contreras, Matthias Prigge 2024. Neuromelanin quantification in stained brain slices of the Locus coeruleus . **protocols.io** https://dx.doi.org/10.17504/protocols.io.5qpvo32y9v4o/v1

Protocol status: Working
We use this protocol and it's working

ABSTRACT

This protocol describes an automatizable pipeline for cell counting based on fluorescent stainings with the help of CellPose. Additionally, it describes the entire generation process of plots that show the intensity of neuromelanin per cell based on bright-field images. The protocol includes Fiji macro codes for data analysis and an R code for plot generation. This pipeline has been used for the analysis of locus coeruleus data.

MATERIALS

- Bright-field images of neuromelanin positive regions, and fluorescent staining images of TH (tyrosine hydroxylase)
- CellPose (2.2.3)
- ImageJ/Fiji (1.54f)
- Excel or LibreOffice Calc
- RStudio (2023.09.1) and R (4.2.1)

Created: Nov 17, 2023

Last Modified: Jan 31, 2024

PROTOCOL integer ID: 91088

Keywords: cell pose, cell segmentation. neuromelanine, qunatification

Funders Acknowledgement:
ASAP
Grant ID: 020505

Image optimization and cropping

1 Start with images of 35µm thick slices of one z-plane and merged channels. In case of several z-planes, create a Maximum Intensity or a Sum Slices projection in ImageJ/Fiji.

2 Separate the channels and **save each channel with an informative title in TIFF format (eg. channel_mouse-id_slice-info_side.tif)**. It is useful to use the same type of separator between each piece of information.

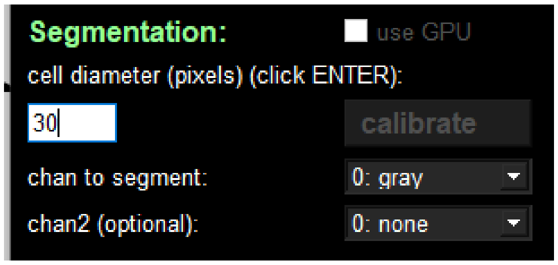
3 Crop the area that you would like to analyze.

Fluorescent image analysis with CellPose

4 Install CellPose following these steps: https://pypi.org/project/cellpose/. You can use Nvidia GPUs to make the software more efficient.

5 Open CellPose and drag a fluorescent image over the gui to open it.

6 Estimate your average cell size; you can do it with the help of Fiji.



cell diameter (pixels) (click ENTER): 30

chan to segment: 0: gray

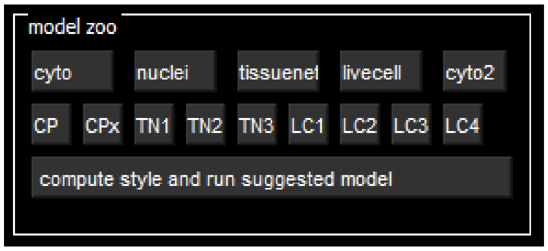
chan2 (optional): 0: none

Press enter. On the bottom left of the picture you will see a pink circle which has the diameter that you have specified.

7 Choose which channel to segment. (Gray works for any image with only one channel.)

8 Find a reliable model.

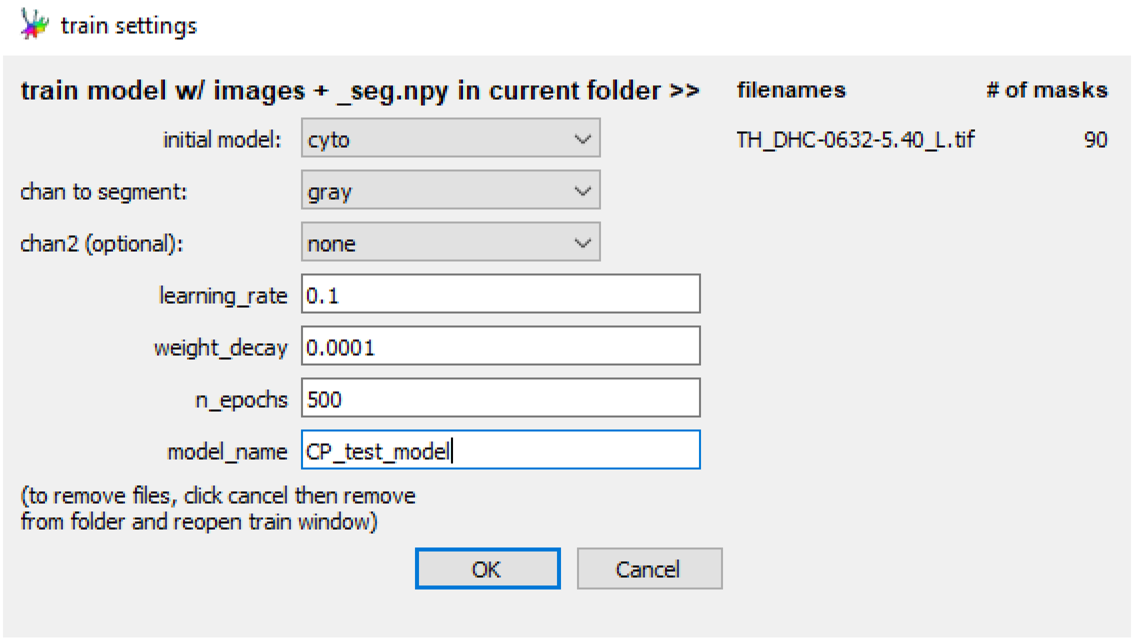
8.1 **Try pretrained models**. You can do it by clicking on any of these. The compute style and run suggested model will suggest a model for you. Try cyto2 - this is probably the most robust model so far.



8.2 If the pretrained models do not give a reliable segmentation, choose the pretrained model that works the best and train it to become better. You can remove unnecessary masks by holding Ctrl and clicking on the mask. If you want to draw more masks, go to the edge of a cell, right click, and then you can draw the outline of the cell. The resulting image+masks will be automatically saved in the folder of the image in an *.npy format.

8.3 Create your training dataset. You should have at least 20 reliable *.npy files saved in the same folder.

8.4 Train a model (Ctrl+T). You have to choose the initial (pretrained) model, you can specify the learning rate, the weight decay and the number of epochs. ~500 epochs can give a reliable, but not overfitted model. Give a name to your network. On the right, you can see all the *.npy files in your folder which will be used as the training dataset.


train settings

train model w/ images + _seg.npy in current folder >>

initial model:	cyto
chan to segment:	gray
chan2 (optional):	none
learning_rate	0.1
weight_decay	0.0001
n_epochs	500
model_name	CP_test_model

filenames	# of masks
TH_DHC-0632-5.40_L.tif	90

(to remove files, click cancel then remove from folder and reopen train window)

9 **Segment your images.** That is: drag each image to the gui, run the model that you chose, take note of the number of identified ROIs and save your masks. Save them in one folder in PNG format for further analysis. It is possible to automate this step in

Software	
Cell Profiler	NAME

, but only with pretrained models.

Neuromelanin quantification per cell (based on TH staining and bright-field)

10 Save your bright-field images and the CellPose masks based on the TH staining images in two separate folders. Make sure that the two folders contain the same amount of images, and the images are in the correct order (so that the first bright-field image corresponds to the first CellPose mask, the 2nd to the 2nd and so on).

11 Open ImageJ/Fiji. Go to Analyze > Set measurements, and untick every option, except for Mean gray value.

12 Then go to Plugins > New > Macro and copy the following macro:

```
function roi_per_cell(input, filename){
        open(input + filename);
        setThreshold(0, 0);
        setOption("BlackBackground", false);
        run("Convert to Mask");
        run("Invert LUT");
        run("Watershed");
        run("Analyze Particles...", "add");
        run("Select All");
        close();
}

function multimeasure(input, filename){
        open(input + filename);
        run("8-bit");
        run("Grays");
        run("Select All");
        roiManager("Multi Measure");
        roiManager("Deselect");
        roiManager("Delete");
        close();
}

input1 = "C:/Documents/histology/masks/TH_masks/" // Change this path. This
is the path to the TH mask folder.
input2 = "C:/Documents/histology/images/BF/" // Change this path. This is
the path to the bright-field image folder.
output = "C:/Documents/histology/results/" // Change this path. This is the
path to the folder where you store the results.

list1 = getFileList(input1);
list2 = getFileList(input2);
for (i = 0; i < list1.length; i++){
        roi_per_cell(input1, list1[i]);
                multimeasure(input2, list2[i]);
                String.copyResults();
                string = String.paste();
                File.append(string, output + "Results.csv");
}
```

This macro loops over the two folders and saves the mean gray value of each TH+ ROI identified by CellPose on each bright-field image. Each line corresponds to one image, and each cell is an ROI. The values are distributed between 0 and 255, where 255 is white, and 0 is black.
If you use Excel to open the *.csv file, you might have to set the separator to be a semicolon.

13 Name every column starting with v1.

14 Add a few columns to the beginning of the csv. Copy and paste all your image paths from one folder (select all files in your file manager, and copy path) and convert these paths into useful columns. Ctrl+H helps to delete unnecessary information (eg. folder info). In Excel or in LibreOffice, go to Data and Text to columns, and choose the separator which you use in your file names. Name your columns. If it is necessary, add a column which contains information about the experimental group.

15 In ImageJ/Fiji, Go to Analyze > Set measurements, and untick every option, except for Modal gray value.

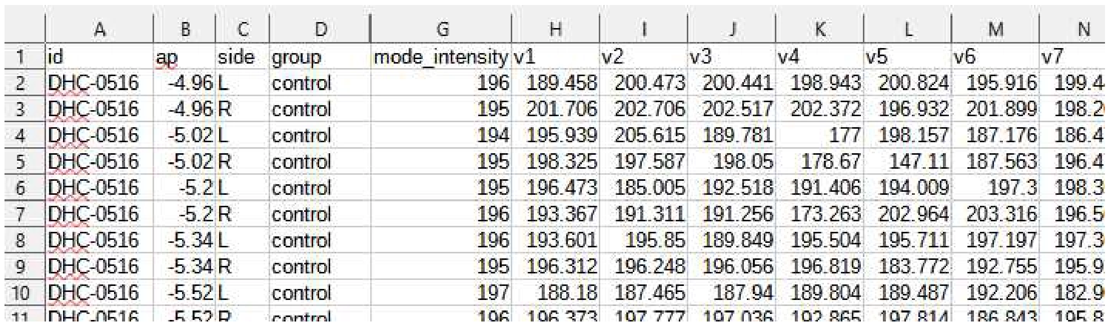
16 Then go to Plugins > New > Macro and copy the following macro:

```
function measure1(input, filename){
        open(input + filename);
        setOption("ScaleConversions", true);
        run("8-bit");
        run("Select All");
        run("Measure");
        close();
}

input = "C:/Documents/histology/images/BF/" // Change this path. This is the
path to the bright-field image folder.

list = getFileList(input);
for (i = 0; i < list.length; i++){
        measure1(input, list[i]);
}
```

Run this macro, and copy the results into one empty column of your csv. This is the modal gray value of each bright-field image, which is important for the normalization of the data.
At this point, your csv should look approximately like this:



	A	B	C	D	G	H	I	J	K	L	M	N
1	id	ap	side	group	mode_intensity	v1	v2	v3	v4	v5	v6	v7
2	DHC-0516	-4.96	L	control	196	189.458	200.473	200.441	198.943	200.824	195.916	199.4
3	DHC-0516	-4.96	R	control	195	201.706	202.706	202.517	202.372	196.932	201.899	198.2
4	DHC-0516	-5.02	L	control	194	195.939	205.615	189.781	177	198.157	187.176	186.4
5	DHC-0516	-5.02	R	control	195	198.325	197.587	198.05	178.67	147.11	187.563	196.4
6	DHC-0516	-5.2	L	control	195	196.473	185.005	192.518	191.406	194.009	197.3	198.3
7	DHC-0516	-5.2	R	control	196	193.367	191.311	191.256	173.263	202.964	203.316	196.5
8	DHC-0516	-5.34	L	control	196	193.601	195.85	189.849	195.504	195.711	197.197	197.3
9	DHC-0516	-5.34	R	control	195	196.312	196.248	196.056	196.819	183.772	192.755	195.9
10	DHC-0516	-5.52	L	control	197	188.18	187.465	187.94	189.804	189.487	192.206	182.9
11	DHC-0516	-5.52	R	control	196	196.373	197.777	197.036	192.865	197.814	186.843	195.8

In this example, ap refers to the approximate anterioposterior position of each slice calculated from Bregma.

17

Software

RStudio NAME

RStudio Team (2020) DEVELOPER

Open RStudio, create a new R notebook and copy the following code to visualize the distribution of the ROI intensities. This code presupposes a dataset with two experimental groups ('side' variable) and four different time points ('time' variable). However, you can adjust this code to your needs, for example, if you do not have a 'time' variable, you can remove it from the code.

~~~
---
title: "Ray fish plots"
output: html_notebook
---

Install the necessary packages.
```{r}
install.packages("tidyverse")
library(tidyverse)
install.packages("janitor")
library(janitor)
install.packages("rstatix")
library(rstatix)
install.packages("ggpubr")
library(ggpubr)
install.packages("svglite")
library(svglite)
```
Read your csv.
```{r}
df<-read.csv("C:/Documents/histology/results/Results.csv") # Change this
path to point to your dataset! Make sure your csv has the same structure as
the csv showed in step 16 (and possibly the same column names).
df <- df %>% clean_names() # This command makes all your variable names
lowercase, and each word will be separated with an underscore.
df<-df %>% convert_as_factor(time, id, ap, side) # Convert your factor
variables to factor type.
df$time<-factor(df$time, levels = c("1.5m", "3m","4m","7m"), labels = c("1.5
months", "3 months", "4 months", "7 months")) # You might not need this
line. Here I rename my time points. If your time points appear on the plot
in a weird order, you can also mend the problem with this command.
df$side<-factor(df$side, levels = c("R", "L"), labels = c("Tyr-", "Tyr+")) #
This line is also optional. I preferred calling the two sides Tyr- and Tyr+.
df$mode_norm<-df$mode_intensity-min(df$mode_intensity) # With this command
you calculate the difference between the modal intensity of each image and
the minimum modal intensity in your dataset. This is important in order to
normalise your data.
df # Visualise your data frame.
```
```{r}
long <- df %>% # Convert your data frame to a long format.
  pivot_longer(
    cols = "v1":"v153", # Make sure to change "v153" to the number of
columns you have.
~~~

```
    names_to = "num",
    values_to = "avg_int"
)
long$norm_norm_int<-long$avg_int-long$mode_norm # With this command you
normalise everything to the lowest modal intensity.
long<-long %>% filter(!is.na(norm_norm_int)) # This line is completely
unnecessary. If you do not run it, you get a warning when you create the
plot, but that is ok.
long # Visualise your long format data frame.
```

~~~
Create the plot and save it in *.svg format.
```{r}
p<-long %>% ggplot(aes(x = side, y = norm_norm_int, fill = side)) + # Define
x, y and color code. x is the variable which defines the two experimental
groups.
  geom_violin(aes(color = side), alpha = 0.4, size = 3, scale = "count") + #
Add ray fish.
  geom_jitter(aes(color = side), size = 1, alpha =1, position =
position_jitterdodge(jitter.width = NULL,jitter.height = 0, dodge.width =
.9)) + # Add jitter. Each dot corresponds to an ROI/cell.
  scale_color_manual(values = c("skyblue3","gray7"),aesthetics =
c("colour","fill")) + # Define colors you would like to use. Search for
RStudio colors or give the hexadecimal code. You need as many colors as the
number of levels your "side" variable has.
  theme(panel.background = element_rect(fill = 'white', colour =
'white'),legend.position = "right", axis.line = element_line(colour =
'gray7', linewidth = 2.5),  panel.grid = element_blank())+ theme(text =
element_text(size = 55), axis.ticks.length=unit(.25, "cm"), axis.ticks =
element_line(linewidth = 2.5), plot.title = element_text(hjust = 0.5)) + #
This is just aesthetic stuff. If you want to remove the legend, set its
position to "none".
  ggtitle("") + # Give title.
  xlab("Time after injection") + # Name x axis.
  ylab("Average color intensity\n(normalized)") + # Name y axis.
  facet_grid(cols = vars(time)) # Define facets. Remove this if you do not
have a time variable !
ggsave("ray_fish.svg", width = 30, height = 10) # Save the plot as SVG.
Adjust the width to your needs.
p
dev.off()
~~~
You can find your plot in the following folder:
```{r}
getwd()
```
```
```

18 Calculate statistics.