PyF2F_Estimate_Distances_Walkthrough

June 7, 2023

PyF2F - Estimate Distances Notebook

This notebook follows an example of PyF2F's workflow from start to finish to estimate the distance between two fluorophores labeling the two subunits of the exocyst complex: Exo70 (C-terminal) and Sec5 (C-terminal) (Picco et al., 2017)

Workflow: a set of ~ 30 two channel images (corresponding to the two labelled subunits) are processed and analysed to estimate the distance between the two fluorophores. For more information see our paper (*link*).

0.0.1 Scientific IPython Setup

Before starting, we need to : 1. Scientific IPython Setup 2. Set parameters to run the distance estimation workflow 3. Set paths to working directories

1. Scientific IPython Setup

Load some required scientific Python libraries:

[1]:
```
%load_ext autoreload
%autoreload 2
import os
import glob
import shutil
import sys
import time
import pandas as pd
import numpy as np
import trackpy as tp
import pims
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import tkinter as tk
from tkinter import filedialog
from pymicro.view.vol_utils import compute_affine_transform
from matplotlib import rcParams
from skimage import io, util
from skimage import data
```

```
from scipy import stats
from scipy import spatial
```

as well as PyF2F's custom functions:

[2]:
```
# Append PyF2F scripts path and import functions
sys.path.append('/'.join(os.getcwd().split("/")[:-1]) + '/scripts/')
from functions import *
from detect_beads import *
%aimport detect_beads
%aimport functions
```

```
Using TensorFlow backend.
```

```
WARNING:tensorflow:From /home/altair/anaconda3/envs/py37/lib/python3.7/site-
packages/tensorflow/python/compat/v2_compat.py:107: disable_resource_variables
(from tensorflow.python.ops.variable_scope) is deprecated and will be removed in
a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

2. Parameters

Now, we have to set the **parameters** to run the image registration workflow:

[3]:
```
###############
# PARAMETERS
###############
# Pixel size of the camera
px_size = 64.5              # Zyla camera (64.5 nm/px) // Prime camera (110 nm/
 ↪px)
####################
# 1. Pre-processing
####################
rbr = 70                   # BGN Subtraction (rolling ball radius)
mfr = 10                    # BGN Subtraction (median filter radius)
###################
# Spot Detection
###################
sdd = 11                    # Spot Detection (diameter of the spot (~ 2·PSF))
sdpc = 99.7                # Spot Detection (select spot intensities above this␣
 ↪percentile)
sdl = 2                    # Spot Detection (max separation to link c1-c2 spots)

##############################
# YC Segmentation
# (YeastSpotter parameters)
##############################
```

```
rescale = False             # Set to true to rescale the input images to reduce␣
 ↪segmentation time
scale_factor = 2            # Factor to downsize images by if rescale is True
save_preprocessed = True    # save preprocessed images as input to neural network
save_compressed = False     # Set to true to save a compressed RLE version of the␣
 ↪masks for sharing
save_masks = True           # Set to true to save the full masks
verbose = True              # print out its segmentation progress as it proceeds
output_imagej = False       # Set to true to output ImageJ-compatible masks
save_contour = True         # Save contour images
save_contour_mod = True     # Save contour modified images

####################
# Spot Selection
####################
ccmd = 13                    # Cell Contour (max distance)
cnmd = sdd - 1              # Closest neighbour (min distance)
kde = 0.5                   # 2D-KDE cutoff (density prob. between 0 - 1)
gauss = 0.35                # gaussian cutoff (R² between 0 - 1)
rjlw = 0                    # reject distances lower thant this (in nm)

####################
# MLE
####################
mle_cutoff = 2/3            # % of distance distribution assumed to be ok.
```

3. Set paths to your working directories

The last thing to go is setting the paths to the working directories. We advise to create a directory to work with. In this example we create a directory called **Exo70_Sec5**. Then we should indicate where are located 1) Beads for registration, 2) Beads for test, and 3) Sample images (PICT images). The following cell takes care of asking us to select:

- **Working directory**
- Directory with beads images to create the registration map (**Beads_reg**)
- Directory with beads images to calculate the error of registration (**Beads_test**)
- Directory with sample images (**PICT images**)

[4]:
```
# Select working directories:
root = tk.Tk()
root.withdraw()
working_directory = filedialog.askdirectory(title="Select Working Directory") +␣
 ↪"/"
path_beads_reg = filedialog.askdirectory(title="Select Directory Beads␣
 ↪Registration") + "/"
path_beads_test = filedialog.askdirectory(title="Select Directory Beads Test")␣
 ↪+ "/"
```

```
sample_directory = filedialog.askdirectory(title="Select PICT Images␣
 ↪Directory") + "/"

# Create directories for output and organize working directory
# Input Dir
path_to_input = working_directory + "input/"
if not os.path.exists(path_to_input):
    os.mkdir(path_to_input)
# Output Dir
path_to_output = working_directory + "output/"
if not os.path.exists(path_to_output):
    os.mkdir(path_to_output)
# Output Reg Beads Dir
path_output_reg = path_to_input + "output_reg/"
if not os.path.exists(path_output_reg):
    os.mkdir(path_output_reg)
# Output Test Beads Dir
path_output_test = path_to_input + "output_test/"
if not os.path.exists(path_output_test):
    os.mkdir(path_output_test)
# PICT images directory
path_to_pict_images = path_to_input + "pict_images/"
if not os.path.exists(path_to_pict_images):
    os.mkdir(path_to_pict_images)
# Copy PICT images to the defined directory path_to_input + "pict_images/"
if len(glob.glob(path_to_pict_images + "*.tif")) == 0:
    for pict_image in glob.glob(sample_directory + "*.tif"):
        img_name = pict_image.split("/")[-1]
        shutil.copyfile(pict_image, path_to_pict_images + img_name)

###################
# PATHS TO OUTPUT
###################
path_to_pp = path_to_output + "images/"             # To save pre-processed␣
 ↪images
path_to_spot_detection = path_to_output + "spots/"  # To save Spot Detection␣
 ↪results in CSV format
path_to_results = path_to_output + "results/"       # To save relevant result␣
 ↪files in CSV format
path_to_figures = path_to_output + "figures/"       # To save figures in PDF,␣
 ↪PNG and HTML format
path_to_segment = path_to_output + "segmentations/"       # To save␣
 ↪segmentations of cells
```

[ ]:
```
print("Your Paths: \n"
          " -Working directory: {}\n"
          " -Beads_Reg directory: {}\n"
```

```
          " -Beads_Test directory: {}\n"
          " -PICT images directory: {}\n"
          " -Output directory: {}\n".format(working_directory,
                                            path_beads_reg,
                                            path_beads_test,
                                            path_to_pict_images,
                                            path_to_output))
```

Now we are ready to run the PyF2F workflow to **estimate the distance** between the **Exo70-mCherry** and the **Sec5-GFP** tags. Notice that all the results will be saved in the "output/" directory.

Step 1. Calculate Regitration Map

Get two channels (c1-c2) coordinates from the REF set of beads to calculate the registration map that will be used to register the two channels. If the coordinates have been previously calculated, then are loaded from the saved CSV files in the "output_reg" and "output_test" directories. This is the same protocol explained in the **PyF2F Image Registration** notebook.

[5]:
```
######################
# 1. Get Coordinates
######################
# Beads Params
beads_head = "beads_*.tif"      # Beads_head (pattern in bead images)
px_size = 64.5                   # Zyla camera (64.5 nm/px) // Prime camera␣
 ↪(110 nm/px)
spot_diameter = 11               # spot detection: diameter of spots in px
percentile = 99.8               # spot detection: sort spots below this␣
 ↪percentile of intensity
min_mass = 0.01                 # spot detection: sort spots with a mass above␣
 ↪this threshold (range 0-1).
max_mass = 0.95                 # spot detection: sort spots with a mass below␣
 ↪this threshold (range 0-1).
max_displacement=1              # linking: link spots from ch1-ch2 channels␣
 ↪separated by this cutoff in px
search_range = 2000 // px_size  # local registration: max distance in px for␣
 ↪nearest-neighbour search
min_fiducials = 100             # local registration: minimum number of␣
 ↪fiducial markers to correct locally

# Check if spots have been previously detected and saved in
# the "path_output_reg" and "path_output_test" directories
if os.path.exists(path_output_reg + "/coords_W1.csv") and os.path.
 ↪exists(path_output_reg + "/coords_W2.csv"):
    c1_ref = np.loadtxt(path_output_reg + "/coords_W1.csv", delimiter=",")
    c2_ref = np.loadtxt(path_output_reg + "/coords_W2.csv", delimiter=",")
    c1_test = np.loadtxt(path_output_test + "/coords_W1.csv", delimiter=",")
    c2_test = np.loadtxt(path_output_test + "/coords_W2.csv", delimiter=",")
```

```
else:
    # Get coordinates from beads for registration
    c2_ref, c1_ref = get_coords(path_output_reg, path_beads_reg, beads_head,␣
 ↪diameter=spot_diameter, separation=max_displacement, percentile=percentile,␣
 ↪min_mass_cutoff=min_mass, max_mass_cutoff=max_mass, px_size=px_size)
    c2_test, c1_test = get_coords(path_output_test, path_beads_test,␣
 ↪beads_head, diameter=spot_diameter, separation=max_displacement,␣
 ↪percentile=percentile, min_mass_cutoff=min_mass, max_mass_cutoff=max_mass,␣
 ↪px_size=px_size)
```

Step 2. Image Pre-processing and Spot Detection

- Pre-processing: Raw PICT images are background subtracted using the rolling ball algorithm. The uneven illumination of the cytoplasmic background is also smoothed by subtracting the median filter.
- Spot Detection: bright spots corresponding to the mCherry and GFP channels are detected in the two channels separately and then linked with a maximum separation parameter.

[6]:
```
######################
# 2. Pre-processing
######################
pp(path_to_pict_images, path_to_pp, rbr, mfr)

######################
# 3. Spot Detection
######################
spot_detection(path_to_pp, path_to_spot_detection, path_to_results,␣
 ↪path_to_figures,
               particle_diameter=sdd, percentile=sdpc, max_displacement=sdl,␣
 ↪px_size=px_size)
```

```
Total Initial W1 Detected spots: 1760

Total Initial W2 Detected spots: 1846

Total Final Paired spots: 784
```

[7]:
```
############################
# 4. Warping (registration)
############################
local_warping(path_output_reg, path_to_spot_detection, path_to_figures,␣
 ↪path_to_results, pixel_size=px_size)
```

```
/home/altair/anaconda3/envs/py37/lib/python3.7/site-
packages/numpy/lib/histograms.py:906: RuntimeWarning:
```

```
invalid value encountered in true_divide
```

Step 3. Spot Selection

- **Distance to the cell contour**: due to the experimental conditions of PICT, detected spots must be located at the plasma membrane of the cell. In this step, PyF2F utilises YeastSpotter to determine the contour of the cell. Spots are sorted according to a maximum distance to the cell contour.
- **Spot pairs in focus**: for determining the distance between spot detected in the two channels, it is crucial to select only those spots that are in focus. Spot pairs in focus should share the same intensity properties. Here, PyF2F selects spot pairs in focus by selecting those sharing simillar eccentricity (round spots) and second momentum of intensity.
- **Goodness of the gaussian fit**: the intensity profiles of the spot pairs should fit well with a 2D-gaussian function. The goodness of the gaussian fit evaluates the quality of detected spots according to this criteria.

[9]:
```
################################
# 5. Distance to Cell Contour
################################
total_data, seg_selected = main_segmentation(path_to_segment, path_to_pp,␣
 ↪path_to_spot_detection,
                                              path_to_results,path_to_figures,␣
 ↪scale_factor, ccmd,
                                              cnmd, rescale=rescale,␣
 ↪verbose=True, px_size=px_size)
################################
# 6. Spot Pairs in focus
################################
kde_initial, kde_selected = main_kde(path_to_pp, path_to_results,␣
 ↪path_to_figures, kde_cutoff=kde, px_size=px_size)

#################################
# 7. Goodness of the Gaussian Fit
#################################
gauss_initial, gauss_selected = main_gaussian(path_to_results, path_to_pp,␣
 ↪path_to_figures, gaussian_cutoff=gauss, px_size=px_size)
```

```
############################
 Segmentation Pre-processing
############################

Processing image 12 …

        Sorting spots…

Image 12 processed in 0.42 s
```

```
Processing image 10 …
        Sorting spots…
Image 10 processed in 0.391 s
Processing image 06 …
        Sorting spots…
Image 06 processed in 0.532 s
Processing image 03 …
        Sorting spots…
Image 03 processed in 0.434 s
Processing image 16 …
        Sorting spots…
Image 16 processed in 0.413 s
Processing image 02 …
        Sorting spots…
Image 02 processed in 0.392 s
Processing image 05 …
        Sorting spots…
Image 05 processed in 0.387 s
Processing image 14 …
        Sorting spots…
Image 14 processed in 0.394 s
Processing image 11 …
        Sorting spots…
Image 11 processed in 0.403 s
```

```
Processing image 13 …
        Sorting spots…
Image 13 processed in 0.38 s
Processing image 17 …
        Sorting spots…
Image 17 processed in 0.404 s
Processing image 18 …
        Sorting spots…
Image 18 processed in 0.429 s
Processing image 07 …
        Sorting spots…
Image 07 processed in 0.422 s
Processing image 19 …
        Sorting spots…
Image 19 processed in 0.48 s
Processing image 20 …
        Sorting spots…
Image 20 processed in 0.423 s
Processing image 01 …
        Sorting spots…
Image 01 processed in 0.397 s
Processing image 09 …
        Sorting spots…
Image 09 processed in 0.398 s
```

```
Processing image 04 …

        Sorting spots…

Image 04 processed in 0.403 s

Processing image 08 …

        Sorting spots…

Image 08 processed in 0.395 s

Processing image 21 …

        Sorting spots…

Image 21 processed in 0.355 s

Processing image 15 …

        Sorting spots…

Image 15 processed in 0.39 s



Total Percent W1 --> 62.14516235453499 %
Total Percent W2 --> 62.789051916055705 %

Total Paired Percent --> 61.659318619843326 % (483 spots)



####################################
Initializing KDE Selection
########################################



Data collected from Segmentation-selected spots!
Total spots: 483


KDE using Silverman's method for clustering…
```

```
Total Paired Percent --> 40.58 %  == 196 spots


Plotting KDE…

KDE analysis done in 5.588 s

##############################
 Gaussian Fitting Selection
##############################

Processing image 12 …

Image 12 processed in 0.113 s

Processing image 10 …

Image 10 processed in 0.071 s

Processing image 06 …

Image 06 processed in 0.093 s

Processing image 03 …

Image 03 processed in 0.057 s

Processing image 16 …

Image 16 processed in 0.086 s

Processing image 02 …

Image 02 processed in 0.068 s

Processing image 05 …

Image 05 processed in 0.063 s

Processing image 14 …

Image 14 processed in 0.08 s

Processing image 11 …

Image 11 processed in 0.126 s

Processing image 13 …
```

```
Image 13 processed in 0.069 s

Processing image 17 …

Image 17 processed in 0.048 s

Processing image 18 …

Image 18 processed in 0.085 s

Processing image 07 …

Image 07 processed in 0.065 s

Processing image 19 …

Image 19 processed in 0.082 s

Processing image 20 …

Image 20 processed in 0.115 s

Processing image 01 …

Image 01 processed in 0.062 s

Processing image 09 …

Image 09 processed in 0.074 s

Processing image 04 …

Image 04 processed in 0.093 s

Processing image 08 …

Image 08 processed in 0.252 s

Processing image 21 …

Image 21 processed in 0.074 s

Processing image 15 …

Image 15 processed in 0.081 s
```

```
Total Gauss-filtered W1 --> 98.0763416477702 %
Total Gauss-filtered W2 --> 99.56709956709958 %

Total Gauss-filtered --> 97.64344121486978 % == 191


Plotting Gaussian selection…
```

Step 4. Distance Estimation

The last step of the workflow consists in determining the true distance between the two fluorescent labels by maximizing the accuracy and precision. At this point, we assume that the majority of the spot pairs are candidates to be real labels of the exocyst subunits Exo70 and Sec5. However, since it is known that the distance distribution is skewed towards large values, we assume that contaminants (outliers) may lie in that area.

Here, PyF2F performs a **maximum likelihood estimate (MLE)** to estimate the true distance μ between the two fluorophores. The MLE searches for the distance that maximizes the probability of the experimental distribution. Outliers are rejected with a **bootstrap method** (see supplementary information).

[10]:
```
###################################
# 8. MLE and Outlier Rejection
###################################
outlier_rejection(path_to_results, path_to_figures, path_to_pp, mu_ini=None,␣
 ↪sigma_ini=None,
                  reject_lower=rjlw, cutoff=mle_cutoff)
```

```
###################################
Initializing Outlier rejection
###########################################


Choosing distance distribution median and stdev as initial values to start
fitting..

        Initial mu: 35.45798824688147
        Initial sigma: 23.28500020624941

        Starting optimization…

-----OPTIM-----
mu : 0.76 +/- 22.59
sigma : 32.36 +/- 1.14
```

```
Starting search of outliers:

        Number of distances: 191
        Cutoff: Explore 33.33333333333334% of distance distribution
        Reject lower: 0 nm

…

-----OPTIM-----
mu : 0.68 +/- 0.45
sigma : 31.85 +/- 0.89
-----OPTIM-----
mu : 0.76 +/- 51.97
sigma : 31.39 +/- 1.31
-----OPTIM-----
mu : 0.47 +/- 122.09
sigma : 30.97 +/- 1.44
-----OPTIM-----
mu : 0.85 +/- 5.69
sigma : 30.56 +/- 1.2
-----OPTIM-----
mu : 1.04 +/- 132.79
sigma : 30.2 +/- 2.48
-----OPTIM-----
mu : 1.37 +/- 25.92
sigma : 29.86 +/- 1.28
-----OPTIM-----
mu : 7.03 +/- 125.62
sigma : 29.15 +/- 15.31
-----OPTIM-----
mu : 15.02 +/- 23.6
sigma : 27.27 +/- 6.62
-----OPTIM-----
mu : 17.91 +/- 14.64
sigma : 26.07 +/- 5.01
-----OPTIM-----
mu : 20.07 +/- 9.06
sigma : 24.95 +/- 3.62
-----OPTIM-----
mu : 21.52 +/- 7.33
sigma : 24.02 +/- 3.18
-----OPTIM-----
mu : 22.4 +/- 6.26
sigma : 23.31 +/- 3.02
-----OPTIM-----
mu : 23.11 +/- 4.83
sigma : 22.66 +/- 2.49
```

```
-----OPTIM-----
mu : 23.71 +/- 4.3
sigma : 22.05 +/- 2.27
-----OPTIM-----
mu : 24.13 +/- 3.97
sigma : 21.53 +/- 2.22
-----OPTIM-----
mu : 24.5 +/- 3.95
sigma : 21.03 +/- 2.23
-----OPTIM-----
mu : 24.84 +/- 3.47
sigma : 20.52 +/- 2.03
-----OPTIM-----
mu : 25.13 +/- 1.84
sigma : 20.04 +/- 1.17
-----OPTIM-----
mu : 25.35 +/- 2.25
sigma : 19.6 +/- 1.37
-----OPTIM-----
mu : 25.51 +/- 2.67
sigma : 19.21 +/- 1.85
-----OPTIM-----
mu : 25.57 +/- 2.81
sigma : 18.9 +/- 1.66
-----OPTIM-----
mu : 25.63 +/- 2.43
sigma : 18.6 +/- 1.51
-----OPTIM-----
mu : 25.66 +/- 2.65
sigma : 18.33 +/- 1.57
-----OPTIM-----
mu : 25.68 +/- 2.16
sigma : 18.07 +/- 1.44
-----OPTIM-----
mu : 25.69 +/- 2.28
sigma : 17.8 +/- 1.42
-----OPTIM-----
mu : 25.71 +/- 2.26
sigma : 17.53 +/- 1.64
-----OPTIM-----
mu : 25.72 +/- 2.21
sigma : 17.25 +/- 1.56
-----OPTIM-----
mu : 25.72 +/- 1.98
sigma : 17.0 +/- 1.32
-----OPTIM-----
mu : 25.69 +/- 1.98
sigma : 16.77 +/- 1.36
```

```
-----OPTIM-----
mu : 25.67 +/- 1.93
sigma : 16.54 +/- 1.33
-----OPTIM-----
mu : 25.61 +/- 1.9
sigma : 16.37 +/- 1.32
-----OPTIM-----
mu : 25.55 +/- 1.94
sigma : 16.18 +/- 1.32
-----OPTIM-----
mu : 25.48 +/- 1.89
sigma : 16.03 +/- 1.28
-----OPTIM-----
mu : 25.4 +/- 1.86
sigma : 15.87 +/- 1.27
-----OPTIM-----
mu : 25.33 +/- 1.83
sigma : 15.72 +/- 1.34
-----OPTIM-----
mu : 25.25 +/- 1.81
sigma : 15.57 +/- 1.24
-----OPTIM-----
mu : 25.17 +/- 1.91
sigma : 15.42 +/- 1.39
-----OPTIM-----
mu : 25.09 +/- 2.06
sigma : 15.28 +/- 1.15
-----OPTIM-----
mu : 25.01 +/- 1.66
sigma : 15.13 +/- 1.18
-----OPTIM-----
mu : 24.93 +/- 1.72
sigma : 14.98 +/- 1.19
-----OPTIM-----
mu : 24.85 +/- 1.7
sigma : 14.83 +/- 1.17
-----OPTIM-----
mu : 24.77 +/- 1.65
sigma : 14.67 +/- 1.12
-----OPTIM-----
mu : 24.69 +/- 1.64
sigma : 14.51 +/- 1.1
-----OPTIM-----
mu : 24.61 +/- 1.49
sigma : 14.37 +/- 1.1
-----OPTIM-----
mu : 24.52 +/- 1.58
sigma : 14.23 +/- 1.12
```

```
-----OPTIM-----
mu : 24.43 +/- 1.58
sigma : 14.12 +/- 1.1
-----OPTIM-----
mu : 24.33 +/- 1.82
sigma : 14.0 +/- 1.11
-----OPTIM-----
mu : 24.24 +/- 1.58
sigma : 13.88 +/- 1.1
-----OPTIM-----
mu : 24.14 +/- 1.57
sigma : 13.77 +/- 1.09
-----OPTIM-----
mu : 24.03 +/- 1.57
sigma : 13.68 +/- 1.09
-----OPTIM-----
mu : 23.93 +/- 1.55
sigma : 13.57 +/- 1.09
-----OPTIM-----
mu : 23.83 +/- 1.39
sigma : 13.47 +/- 1.05
-----OPTIM-----
mu : 23.73 +/- 1.53
sigma : 13.36 +/- 1.06
-----OPTIM-----
mu : 23.63 +/- 1.52
sigma : 13.26 +/- 1.06
-----OPTIM-----
mu : 23.53 +/- 1.51
sigma : 13.16 +/- 1.06
-----OPTIM-----
mu : 23.42 +/- 2.35
sigma : 13.07 +/- 1.44
-----OPTIM-----
mu : 23.32 +/- 1.5
sigma : 12.97 +/- 1.05
-----OPTIM-----
mu : 23.21 +/- 1.49
sigma : 12.87 +/- 1.04
-----OPTIM-----
mu : 23.11 +/- 1.46
sigma : 12.77 +/- 1.03
-----OPTIM-----
mu : 23.01 +/- 1.49
sigma : 12.66 +/- 1.03
-----OPTIM-----
mu : 6.53 +/- 2.79
sigma : -0.07 +/- 2.41
```

```
-----OPTIM-----
mu : 23.28 +/- 1.43
sigma : 12.35 +/- 0.99
-----OPTIM-----
mu : 6.72 +/- 2.87
sigma : -0.12 +/- 2.5
/home/altair/PycharmProjects/book_chapter_local_affine/scripts/rnd.py:31:
RuntimeWarning:

invalid value encountered in double_scalars

/home/altair/PycharmProjects/book_chapter_local_affine/scripts/rnd.py:31:
RuntimeWarning:

invalid value encountered in double_scalars


-----OPTIM-----
mu : 23.54 +/- 1.37
sigma : 12.02 +/- 0.96


#-#-#-#-#-#-#-#-#-#-#-#-#-
Number of initial distances before fitting: 191
Number of distances rejected: 27
Final number of distances: 164
#-#-#-#-#-#-#-#-#-#-#-#-#-


-----BOOTSTRAP-----
max Sh: 0.3195299298005523
mu = 25.724823646983847 +/- 2.2062761458403983
sigma = 17.249397383248155 +/- 1.5649665089945184
n = 164
min Sh: 0.0027042699045502303
mu = 23.007374240184536 +/- 1.4894051858617972
sigma = 12.664748099097936 +/- 1.033125408154926
```

0.1 END