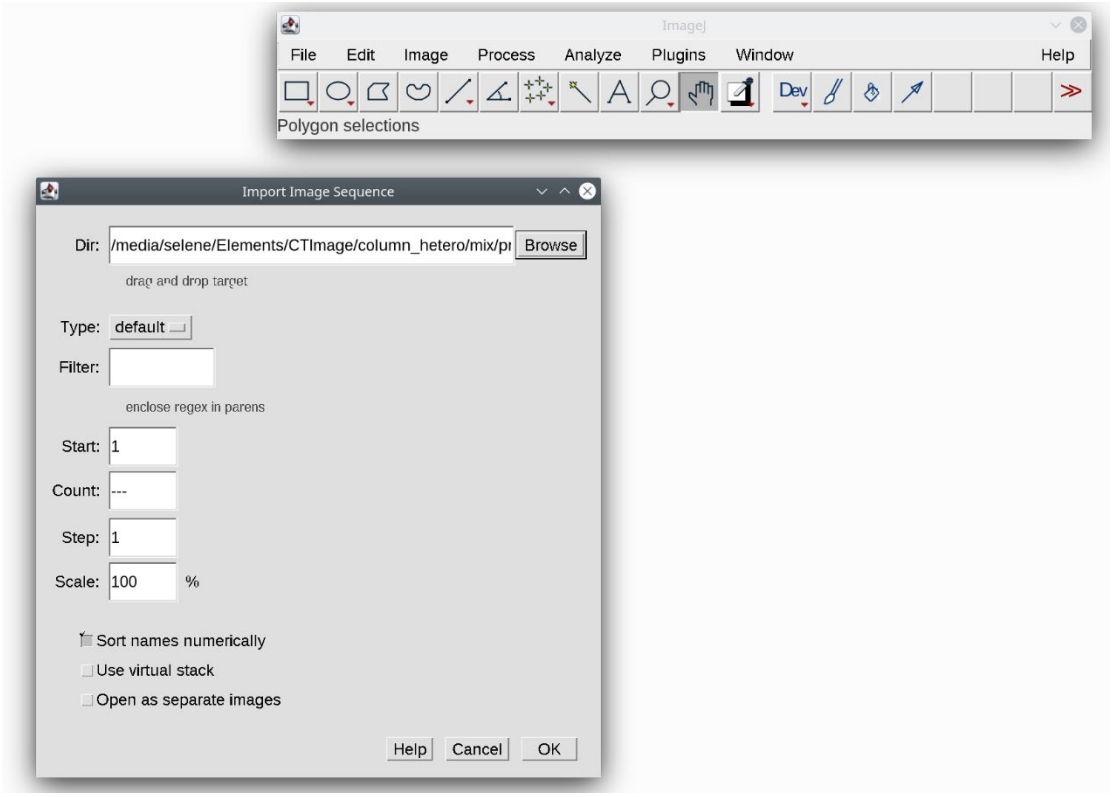
简易 ImageJ 教程 1

1. 打开 ImageJ 的命令

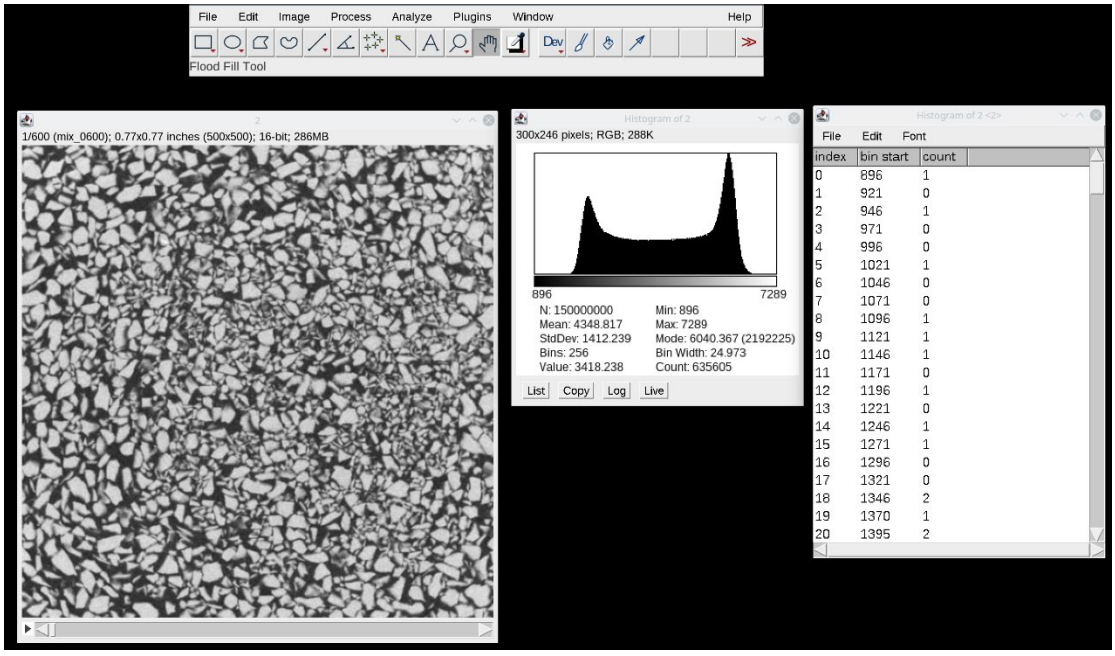
命令行运行 imagej，如果嫌字体太小，运行 GDK_SCALE=2 imagej

2. Histogram 数据输出

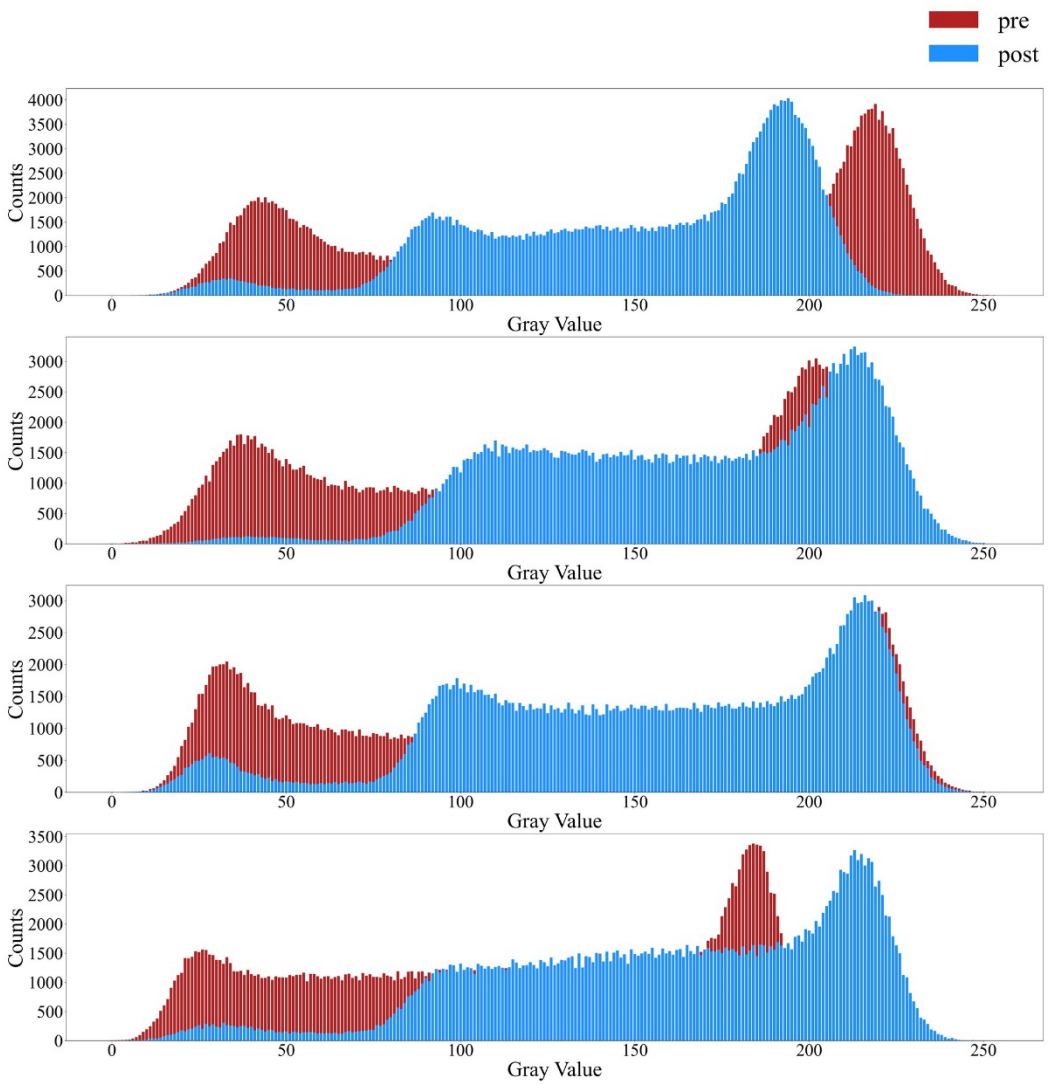
File → Import → Image Sequence，browse 找到图像切片所在的文件夹，然后导入



Analyze → Histogram



File → save as xxx.csv
Excel 或者 python 画图



Python 画图代码 1：

```
import matplotlib.pyplot as plt
import matplotlib.font_manager as font_manager
import numpy as np
import xlrd

def read_xlrd(excelFile):
    data = xlrd.open_workbook(excelFile)
    table = data.sheet_by_index(0)
    dataFile = []
    for rowNum in range(table.nrows):
        if rowNum > 0:
            dataFile.append(table.row_values(rowNum))
    return dataFile
```

```
value1 = list(range(255))
post1 = list(range(255))
pre1 = list(range(255))

value2 = list(range(255))
post2 = list(range(255))
pre2 = list(range(255))

value3 = list(range(255))
post3 = list(range(255))
pre3 = list(range(255))

value4 = list(range(255))
post4 = list(range(255))
pre4 = list(range(255))

excelFile1 = '垂直/HISTOGRAM.xls'
datafile1 = read_xlrd(excelFile=excelFile1)
for i in range(255):
    value1[i] = datafile1[i][0]
    post1[i] = datafile1[i][1]
    pre1[i] = datafile1[i][2]

excelFile2 = '水平/HISTOGRAM.xls'
datafile2 = read_xlrd(excelFile=excelFile2)
for i in range(255):
    value2[i] = datafile2[i][0]
    post2[i] = datafile2[i][1]
    pre2[i] = datafile2[i][2]

excelFile3 = '混合/HISTOGRAM.xls'
datafile3 = read_xlrd(excelFile=excelFile3)
for i in range(255):
    value3[i] = datafile3[i][0]
    post3[i] = datafile3[i][1]
    pre3[i] = datafile3[i][2]

excelFile4 = '环状/HISTOGRAM.xls'
datafile4 = read_xlrd(excelFile=excelFile4)
for i in range(255):
    value4[i] = datafile4[i][0]
    post4[i] = datafile4[i][1]
    pre4[i] = datafile4[i][2]
```

```
font = font_manager.FontProperties(family='Times New Roman', weight='normal', style='normal', size=36)
fig, ax = plt.subplots(4, 1, figsize=(40, 40))
ax = plt.gca()
plt.subplot(411)
plt.ticklabel_format(style='sci', axis='y')
plt.bar(value1,post1, color="firebrick", label="pre")
plt.bar(value1,pre1, color="dodgerblue", label="post")
plt.xlabel("Gray Value", fontsize=48, fontproperties='Times New Roman')
plt.xticks(fontsize=40, fontproperties='Times New Roman')
plt.yticks(fontsize=40, fontproperties='Times New Roman')
ax.yaxis.major.formatter.set_powerlimits((0,0))
ax.xaxis.major.formatter.set_powerlimits((0,0))
plt.ylabel("Counts", fontsize=48, fontproperties='Times New Roman')
plt.legend(loc='upper left',prop=font)

plt.subplot(412)
plt.ticklabel_format(style='sci', axis='y')
plt.bar(value2,post2, color="firebrick", label="pre")
plt.bar(value2,pre2, color="dodgerblue", label="post")
plt.xlabel("Gray Value", fontsize=48, fontproperties='Times New Roman')
plt.xticks(fontsize=40, fontproperties='Times New Roman')
plt.yticks(fontsize=40, fontproperties='Times New Roman')
ax.yaxis.major.formatter.set_powerlimits((0,0))
ax.xaxis.major.formatter.set_powerlimits((0,0))
plt.ylabel("Counts", fontsize=48, fontproperties='Times New Roman')

plt.subplot(413)
plt.ticklabel_format(style='sci', axis='y')
plt.bar(value3,post3, color="firebrick", label="pre")
plt.bar(value3,pre3, color="dodgerblue", label="post")
plt.xlabel("Gray Value", fontsize=48, fontproperties='Times New Roman')
plt.xticks(fontsize=40, fontproperties='Times New Roman')
plt.yticks(fontsize=40, fontproperties='Times New Roman')
ax.yaxis.major.formatter.set_powerlimits((0,0))
ax.xaxis.major.formatter.set_powerlimits((0,0))
plt.ylabel("Counts", fontsize=48, fontproperties='Times New Roman')

plt.subplot(414)
plt.ticklabel_format(style='sci', axis='y')
plt.bar(value4,post4, color="firebrick", label="pre")
plt.bar(value4,pre4, color="dodgerblue", label="post")
plt.xlabel("Gray Value", fontsize=48, fontproperties='Times New Roman')
plt.xticks(fontsize=40, fontproperties='Times New Roman')
```

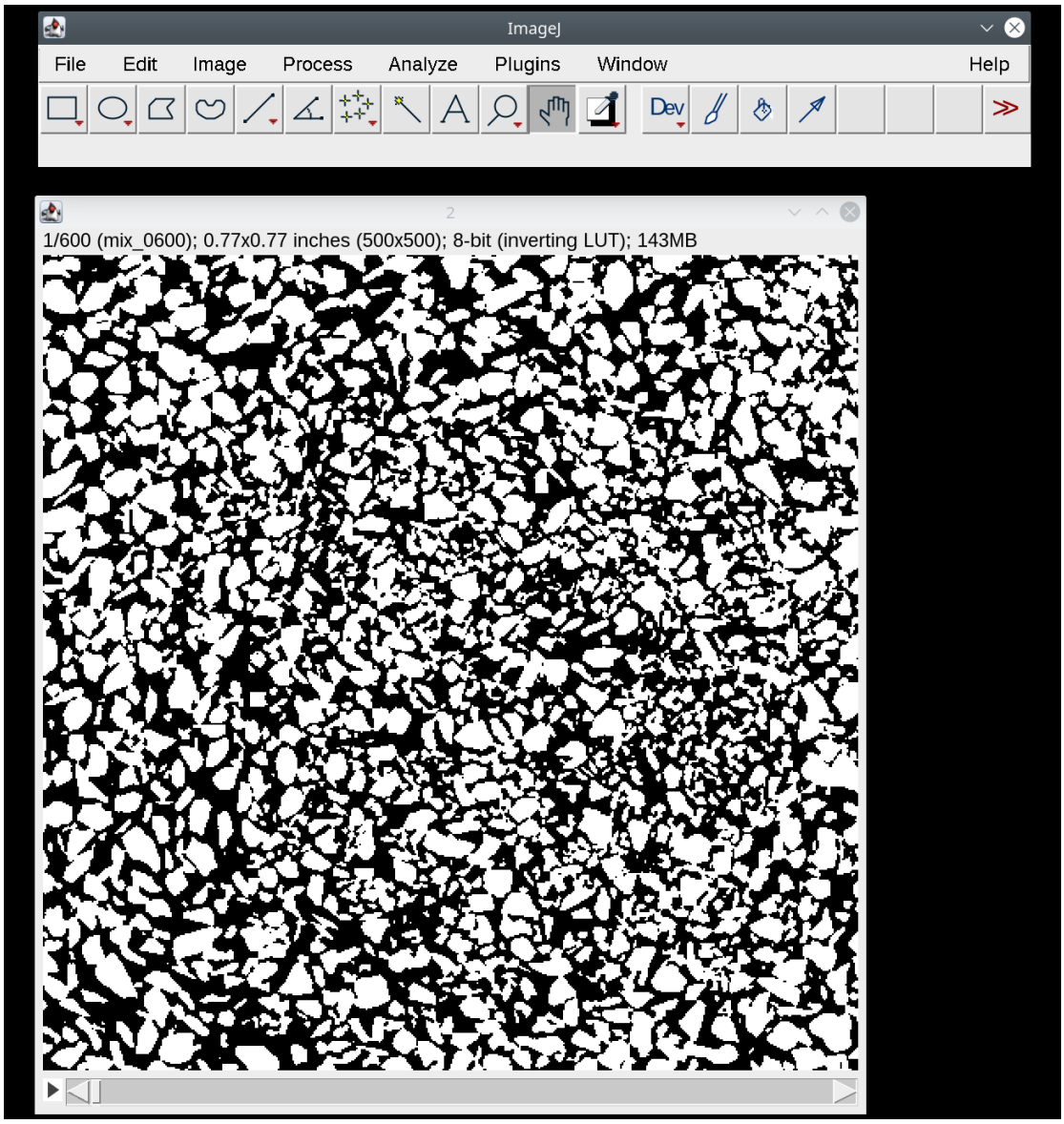
```
plt.yticks(fontsize=40, fontproperties='Times New Roman')
ax.yaxis.major.formatter.set_powerlimits((0,0))
ax.xaxis.major.formatter.set_powerlimits((0,0))
plt.ylabel("Counts", fontsize=48, fontproperties='Times New Roman')

fig.savefig("histobar.png", dpi=300)
```

Python 画图代码 2：

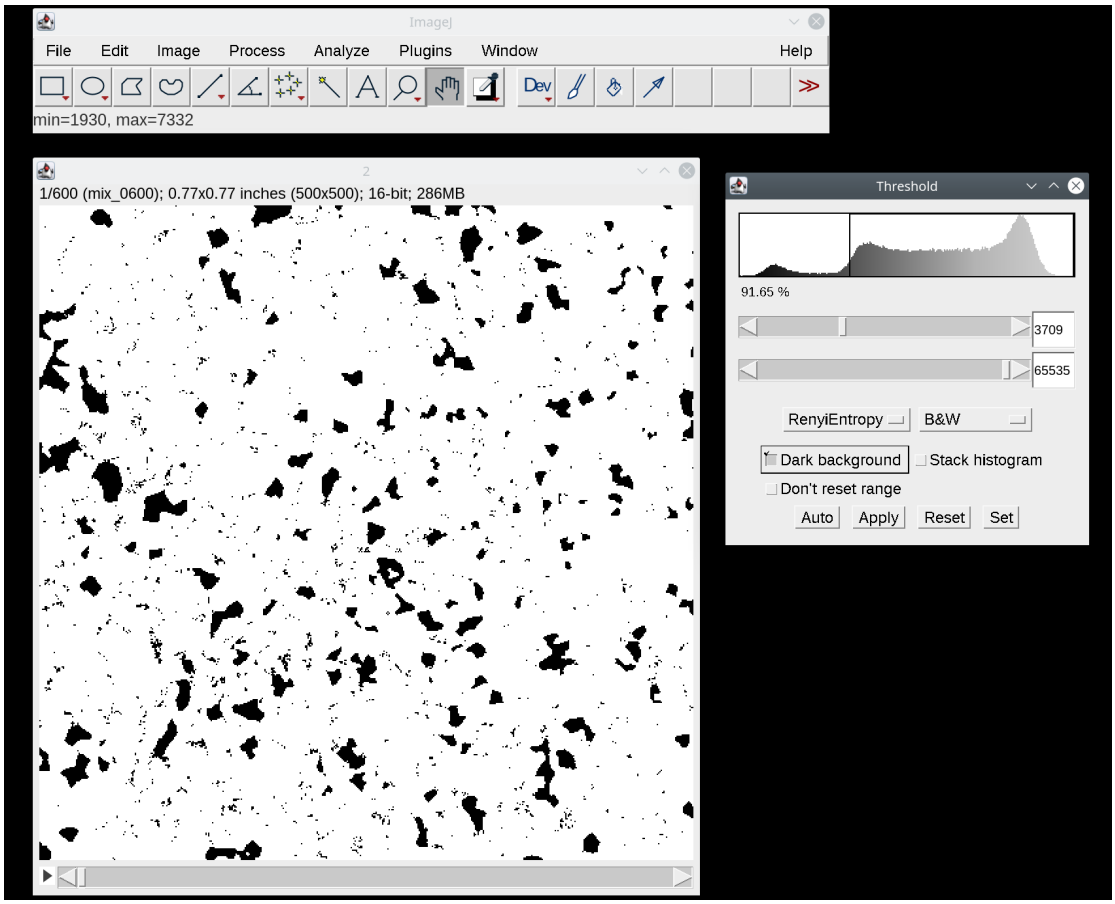
```
import sys
import numpy as np
import skimage.color
import skimage.io
from matplotlib import pyplot as plt

# read image, based on command line filename argument;
# read the image as grayscale from the outset
image = skimage.io.imread(fname='starch/starch.png', as_gray=True)

# display the image
skimage.io.imshow(image)

# create the histogram
histogram, bin_edges = np.histogram(image, bins=256, range=(0, 1))

# configure and draw the histogram figure
plt.figure()
plt.title("Grayscale Histogram")
plt.xlabel("grayscale value")
plt.ylabel("pixels")
plt.xlim([0.0, 1.0])    # <- named arguments do not work here

plt.plot(bin_edges[0:-1], histogram)    # <- or here
plt.show()
#plt.imsave('垂直/histogram-post.jpg')
```

3. Binary image 制作

File → open，导入单张灰度图
如果没有沉淀，process → binary → make binary

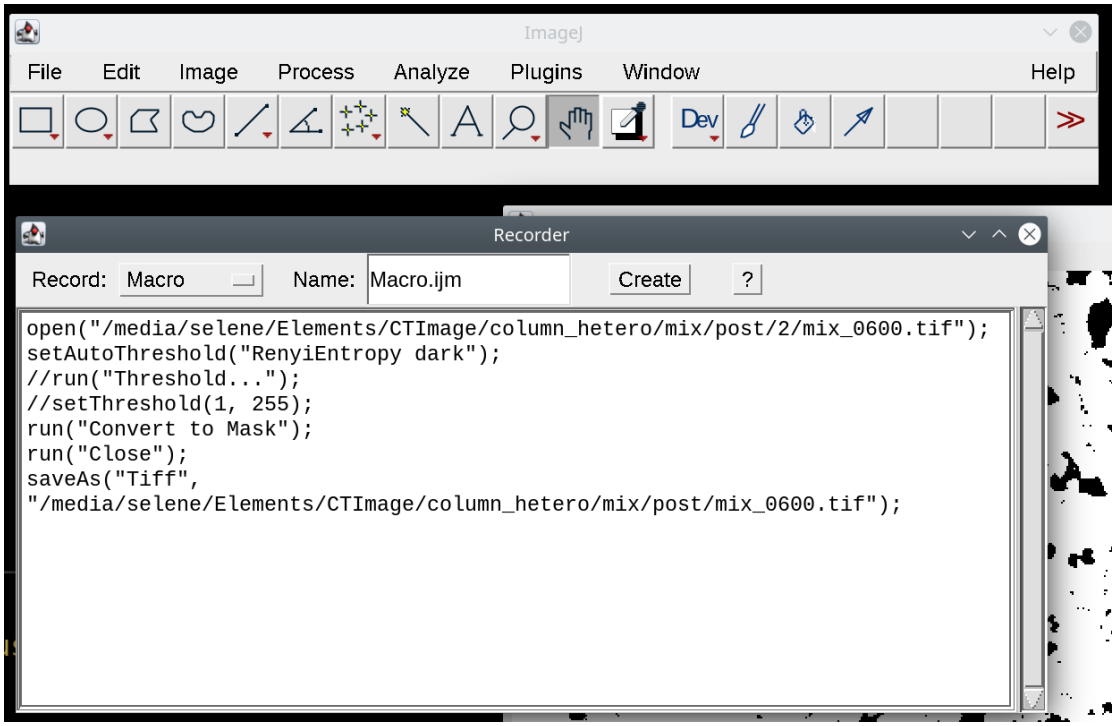


如果有沉淀，需要打开 process → adjust → threshold



4. 批量操作——Macro 文件制作

在不打开任何文件的情况下，Plugins → Macros → record
然后打开一个图片，进行想要的操作，并保存，record 界面会自动生成每步操作对应的命令。
点击 create，弹出 Macro.ijm 对话框，点击 File → save as，保存 Macro.ijm 文件。



创建批量复制 macro 的模板：打开 macro.ijm，可以使用 vim 打开，命令为 vim Macro.ijm，进入 vim 后，通过上下左右键控制光标位置。首先，按一下 i 键，使文本处于编辑模式，然后将所有路径中，具体的文件名替换成 filename。如下图所示：

```
open("/media/selene/Elements/CTImage/column_hetero/mix/post/binary/post_@filename@.tif");
open("/media/selene/Elements/CTImage/column_hetero/mix/pre/binary/pre_@filename@.tif");
imageCalculator("Subtract create", "post_@filename@.tif","pre_@filename@.tif");
selectWindow("Result of post_@filename@.tif");
saveAs("Tiff", "/media/selene/Elements/CTImage/column_hetero/mix/calcite/calcite_@filename@.tif");
```

5. 批量操作——Linux 命令行运行

批量制作 macro 文件：

对于一个叫做的 sand000.tif – sand999.tif 的 image sequence

运行 for i in `seq -f sand%3g.tif 0 999`; do seed -e “s/@filename@/$i/g” Macro.ijm > Macro$i.ijm; done

运行上述命令将会出现很多 macro 文件，并且每个文件都对应相应的一张灰度图。

然后，运行如下命令对所有灰度图进行批量操作：

for i in `seq -f sand%3g.tif 0 999`; do imagej -bMacro$i.ijm; done