```
import matplotlib.pyplot as plt
import matplotlib.patches as mplpatches
import numpy as np
import matplotlib
import argparse
from matplotlib.patheffects import withStroke

#initialize the argument parser and style sheet
parser = argparse.ArgumentParser()
parser.add_argument("--outFile", "-o" ,type=str,action="store",help="output file")
parser.add_argument("--positionFile", "-p" ,type=str,action='store',help="position
file")
parser.add_argument("--cellTypeFile", "-c" ,type=str,action='store',help="cellType
file")
args = parser.parse_args()
outFile=args.outFile
position=args.positionFile
cellType=args.cellTypeFile

plt.style.use("BME163")

#colors
iBlue=(88/255,85/255,120/255)
Grey=(180/255,180/255,180/255)
iGreen=(120/255,172/255,145/255)

#create cell type dict
cellColorDict={"monocyte": Grey,
               "tCell": iGreen,
               "bCell": iBlue}

#Set up figure
figureWidth=5
figureHeight=3
plt.figure(figsize=(figureWidth,figureHeight))

#set up panels

#main panel
panelLeftMain=0.5
panelBottomMain=0.5
panelWidthMain=1.5
panelHeightMain=1.5
panelMain=plt.axes([panelLeftMain/figureWidth,panelBottomMain/
figureHeight,panelWidthMain/figureWidth,panelHeightMain/figureHeight])
panelMain.set_xlim(-30,30)
panelMain.set_ylim(-40,30)
panelMain.set_xticks([-20,0,20])
panelMain.set_yticks([-40,-20,0,20])
panelMain.set_xlabel("tSNE 2") #x-axis labels
panelMain.set_ylabel("tSNE 1") #y-axis labels

#density panel
panelLeftDensity=2.5
panelBottomDensity=0.5
panelWidthDensity=1.5
panelHeightDensity=1.5
panelDensity=plt.axes([panelLeftDensity/figureWidth,panelBottomDensity/
figureHeight,panelWidthDensity/figureWidth,panelHeightDensity/figureHeight])
```

```
panelDensity.set_xlim(-30,30)
panelDensity.set_ylim(-40,30)
panelDensity.set_xticks([-20,0,20])
panelDensity.set_yticks([-40,-20,0,20])
panelDensity.set_xlabel("tSNE 2") #x-axis labels
panelDensity.set_ylabel("tSNE 1") #y-axis labels

#gradient panel
panelGradient=4
panelBottomGradient=1.1
panelWidthGradient=0.1
panelHeightGradient=0.3
panelGradient=plt.axes([panelGradient/figureWidth,panelBottomGradient/
figureHeight,panelWidthGradient/figureWidth,panelHeightGradient/figureHeight])
panelGradient.set_xticks([])
panelGradient.set_yticks([0,101])
panelGradient.yaxis.tick_right()
panelGradient.set_yticklabels(["Min","Max"])
panelGradient.set_ylim(0,101)

#create gradient plot
#heatmap:
viridis5 = (253/255, 231/255, 37/255)
viridis4 = (94/255, 201/255, 98/255)
viridis3 = (33/255, 145/255, 140/255)
viridis2 = (59/255, 82/255, 139/255)
viridis1 = (68/255, 1/255, 84/255)

R1=np.linspace(viridis1[0],viridis2[0],26)
G1=np.linspace(viridis1[1],viridis2[1],26)
B1=np.linspace(viridis1[2],viridis2[2],26)

R2=np.linspace(viridis2[0],viridis3[0],26)
G2=np.linspace(viridis2[1],viridis3[1],26)
B2=np.linspace(viridis2[2],viridis3[2],26)

R3=np.linspace(viridis3[0],viridis4[0],26)
G3=np.linspace(viridis3[1],viridis4[1],26)
B3=np.linspace(viridis3[2],viridis4[2],26)

R4=np.linspace(viridis4[0],viridis5[0],26)
G4=np.linspace(viridis4[1],viridis5[1],26)
B4=np.linspace(viridis4[2],viridis5[2],26)

R=np.concatenate((R1[:-1],R2[:-1],R3[:-1],R4),axis=None)
G=np.concatenate((G1[:-1],G2[:-1],G3[:-1],G4),axis=None)
B=np.concatenate((B1[:-1],B2[:-1],B3[:-1],B4),axis=None)

#for loop for creating the full color map
colormap=[]
for i in range(0,101,1):
    colormap.append((R[i],G[i],B[i]))
#for loop for plotting gradient
for i in range(0,101,1):
    rectangle1=mplpatches.Rectangle((0,i),1,1,
                                    facecolor=colormap[i],
                                    linewidth=0,
                                    edgecolor="black"
    )
```

```
    panelGradient.add_patch(rectangle1)

#read in position data
cellDict={}
for line in open(position):
    splitLine=line.strip().split() #cleans each line, spaces were not /t
    key=splitLine[0] #Cell ID
    value=[float(splitLine[1]), float(splitLine[2])] #Columns 2 and 3 as the value
(tuple)
    cellDict[key]=value #Add to dictionary

#read in cell type data
for line in open(cellType):
    splitLine=line.strip().split()
    key=splitLine[2] #Cell ID
    value=splitLine[1] #Column 2 as the value (cell type)
    if key in cellDict: #Check if the key exists in the dictionary
        cellDict[key].append(value) #Add cell type to dict

#panel plotting and overlap caluclation
msize=4 #marker size
minDist=msize/36
xRange=60 #normalize values to panel x and y limits for overlap calculation
yRange=70
for point1 in cellDict.values(): #nested for loop for points
    overlap=-1 #counter for overlap to map to colormap, starts at -1 due to the
nested for loop
    for point2 in cellDict.values():
        xDist=((point1[0]-point2[0])/xRange)*panelWidthDensity #calc euclidean
distance
        yDist=((point1[1]-point2[1])/yRange)*panelHeightDensity
        distance=(xDist**2 + yDist**2)**0.5
        if distance<minDist: #check if points overlap and adds to the overlap
counter
            overlap+=1
    if overlap>100: #any count over 100 will be set to 100 to avoid index error in
the colormap
        overlap=100

    #plot density plot
    panelDensity.plot(point1[0], point1[1], marker="o",
                      linestyle="",
                      markersize=msize,
                      color=colormap[overlap],
                      markeredgewidth=0.05,
                      alpha=1,
                      )
    panelMain.plot(point1[0], point1[1], marker="o", #plots main panel instead of
using seprate for loop
                     linestyle="",
                     markersize=4,
                     color=cellColorDict[point1[2]], #indexes the color dict
                     alpha=1,
                     markeredgecolor="black",
                     markeredgewidth=0.1)

#Initialize a dictionary to group x and y values by cell type
cellTypeLists = {
    "monocyte": {"x": [], "y": []},
```

```
    "tCell": {"x": [], "y": []},
    "bCell": {"x": [], "y": []}
}

#Group x and y values by cell type
for item in cellDict.values():
    cellType = item[2] #item[2] is the cell type
    if cellType in cellTypeLists: #Creates list of x and y positons for each cell
type
        cellTypeLists[cellType]["x"].append(item[0])
        cellTypeLists[cellType]["y"].append(item[1])

#Calculate medians for each cell type
monoctyeXList = np.median(cellTypeLists["monocyte"]["x"])
monoctyeYList = np.median(cellTypeLists["monocyte"]["y"])
tCellXList = np.median(cellTypeLists["tCell"]["x"])
tCellYList = np.median(cellTypeLists["tCell"]["y"])
bCellXList = np.median(cellTypeLists["bCell"]["x"])
bCellYList = np.median(cellTypeLists["bCell"]["y"])

#Add text with a white border
for item in cellTypeLists:
    xMedian=np.median(cellTypeLists[item]["x"])
    yMedian=np.median(cellTypeLists[item]["y"])
    panelMain.text(
        xMedian, yMedian, item,
        ha="center", va="center",
        path_effects=[withStroke(linewidth=1, foreground="white")], #White border
        color="black" #Text color
    )

#Add text to density plot
panelDensity.text(
        -20.64, -35.1, "Density", ha="center", va="center",color="black")

# save figure
plt.savefig(outFile, dpi=600)
```