

```

#!/usr/local/bin/python

import sys
import operator

# allow +- 4Da when assigning compound labeling
def comp(a,b):
    noise = 4
    return (abs(a-b)<noise)

# allow +- 15Da when assigning protein peak
def compProt(a,b):
    noise = 15
    return (abs(a-b)<noise)

# allow +- 8Da when assigning (non-compound) protein adducts
def compAdd(a,b):
    noise = 8
    return (abs(a-b)<noise)

# each well is processed independetly but requires a reference well
# in which
# there was no compound - typically A01
def
assign_sample(unassigned,well,mw_dict,map_dict,top_spec_dict,base_sp
ectra):

    protein = base_spectra[0][0]
    baseChange = 0
    badWell = 0
    #normalize by counts
    norm_spec = []
    max = top_spec_dict[0][1]
    for n in top_spec_dict:
        # only keep peaks above 10% of the maximum identified peak
        # for the reference well, don't remove adducts smaller than
compound size
        if well=="A01":
            if n[1]/max > 0.1:
                norm_spec.append([n[0],n[1]/max,""])
        else:
            # remove <10% labeling and adducts of less than 100
            if n[1]/max > 0.1 and ( (n[0]-protein)>100 or (n[0]-
protein<15 and n[0]-protein>-15) ):
                norm_spec.append([n[0],n[1]/max,""])
            # detect a possible second major species (other than the
unlabeled protein)
            if n[1]/max > 0.8 and ( (n[0]-protein<100) and (n[0]-
protein>15) ):
                baseChange = n[0]-protein
                # count peaks over 45% to determine if this is a bad
well.
                if n[1]/max > 0.45:
                    badWell = badWell+1

```

```

# assign up to four non-compound adducts from the reference well
adducts = {}
adducts ['adduct1'] = base_spectra[1][0]-protein
adducts ['adduct2'] = base_spectra[2][0]-protein
adducts ['adduct3'] = base_spectra[3][0]-protein
adducts ['adduct4'] = base_spectra[4][0]-protein

if well in map_dict.keys():
    for w in map_dict[well]:
        adducts[w] = mw_dict[w]

# assign protein, single and double modifications
bound=[]
foundNotLabeled = 0
unidentified = 0
doubleLabel = 0
adductConfusion = 0

# first identify the unlabeled protein peak
for n in norm_spec:
    if compProt(n[0],protein):
        n[2] = "protein"
        # If this peak is within +-15Da of base protein assign
it
        # as the new protein mass for this well
        protein = n[0]
        print protein
        foundNotLabeled = 1

# first pass - single modification
for n in norm_spec:
    if not compProt(n[0],protein):
        # if peak corresponds to any of the 4 adducts assign it
        for a in adducts:
            if (a=="adduct1" or a=="adduct2" or a=="adduct3" or
a=="adduct4"):
                if compAdd(n[0],protein+adducts[a]):
                    n[2] = a
                    bound.append(a)
                else:
                    # if the peak doesn't correspond to an adduct,
assign it as compound labeling
                    if comp(n[0],protein+adducts[a]):
                        if n[2]=='': # not adduct
                            n[2] = a
                            bound.append(a)
                        else: # was already assigned as an adduct,
overwrite if >30% labeling
                            # more likely to be the compound -
this is in case some adduct
                            # mass corresponds to one of the
compounds mass
                            if (n[1]>0.3):

```

```

        n[2] = a
        bound.append(a)
    else: # if < 0.3 but no unlabeled
protein detected its still a comp and not an adduct
        if not (foundNotLabeled == 1):
            n[2] = a
            bound.append(a)

# second pass – double modification
for n in norm_spec:
    for a in adducts:
        for b in adducts:
            if comp(n[0],protein+adducts[a]+adducts[b]):
                if n[2]=="":
                    #don't assign two adducts to a single
mass
                    if not ( (a=="adduct1" or a=="adduct2"
or a=="adduct3" or a=="adduct4") and (b=="adduct1" or b=="adduct2"
or b=="adduct3" or b=="adduct4") ):
                        n[2]=a,b
                        if a==b:
                            doubleLabel = 1
                        else: #there are two different combinations
that can explain the weight
                            #if only a single mass explains the
spectrum (PCM-XXXXXXX) disregard
                            if len(n[2])>11: #ugly hack assumes
compound names (PCM-XXXXXXX) are 11 chars long
                                #if one of the previously assigned
compounds was also bound as a single – keep this assignment
                                if not ( (n[2][0] in bound) and (not
((n[2][0]=="adduct1" or n[2][0]=="adduct2" or n[2][0]=="adduct3" or
n[2][0]=="adduct4"))) or (n[2][1] in bound) and (not ((n[2]
[1]=="adduct1" or n[2][1]=="adduct2" or n[2][1]=="adduct3" or n[2]
[1]=="adduct4")))):
                                    #do not assign double adducts
over a previous assignment
                                    if not (((a=="adduct1" or
a=="adduct2" or a=="adduct3" or a=="adduct4") and(b=="adduct1" or
b=="adduct2" or b=="adduct3" or b=="adduct4") ) and ((n[2]
[0]=="adduct1" or n[2][0]=="adduct2" or n[2][0]=="adduct3" or n[2]
[0]=="adduct4") or (n[2][1]=="adduct1" or n[2][1]=="adduct2" or n[2]
[1]=="adduct3" or n[2][1]=="adduct4"))):
                                        n[2]=a,b
                                        if a==b:
                                            doubleLabel = 1

                    print n[0]-protein,"|", "%.2f" % n[1],"|",n[2], "<br>"
                    if ((n[2]=="") and ((n[0]-protein)<400)):
                        unassigned.append([n[0]-protein,n[1]]) # [n[0]-
protein]=n[1]

#assign % labeling
total = 0

```

```

labeling = {}
if well in map_dict.keys():
    for w in map_dict[well]:
        labeling[w]=0;
labeling ['protein']=0

# check for potential adduct confusion
# wells in which an adduct corresponds
# in MW to onw of the compounds will be flagged
if well in map_dict.keys():
    for w in map_dict[well]:
        for ad in ('adduct1','adduct2','adduct3','adduct4'):
            if comp(adducts[ad],mw_dict[w]):
                adductConfusion = ad

for n in norm_spec:
    if not ((n[2]== "") or ('adduct1' in n[2]) or ('adduct2' in
n[2]) or ('adduct3' in n[2]) or ('adduct4' in n[2])):
        total=total+n[1]

#for assignment calculation take into account also unidentified
peaks with significant labeling (>30% of the assigned)
for n in norm_spec:
    if (total > 0) and (n[2]== "") and ( n[1]/total>0.3):
        total=total+n[1]

for n in norm_spec:
    if not (('adduct1' in n[2]) or ('adduct2' in n[2]) or
('adduct3' in n[2]) or ('adduct4' in n[2])):
        if well in map_dict.keys():
            for w in map_dict[well]:
                if w in n[2]:
                    labeling[w] = labeling[w] + n[1]/total
        if n[2]=="protein":
            labeling['protein'] = n[1]/total
        if (total > 0 ) and (n[2]== "") and ( n[1]/total>0.3):
            unidentified = 1

#Produce nice looking HTML tabel with compound figures and %
labeling
for l in labeling:
    if not "protein" in l:
        print "<td>"
        if labeling[l]>=0.7:
            print "<img src='../pngs/'+l+'.png' width=145
border=3 style='border-color:#ff0000'><br>"
        if labeling[l]<0.7 and labeling[l]>=0.5:
            print "<img src='../pngs/'+l+'.png' width=145
border=3 style='border-color:#ff5500'><br>"
        if labeling[l]<0.5 and labeling[l]>=0.3:
            print "<img src='../pngs/'+l+'.png' width=145
border=3 style='border-color:#ffaa00'><br>"
        if labeling[l]<0.3 and labeling[l]>0.0:
            print "<img src='../pngs/'+l+'.png' width=145

```

```

border=3 style='border-color:#ffee00'><br>"
    if labeling[l]==0:
        print "<img src='../pngs/'+l+'.png' width=145"
border=3 style='border-color:#ffffff'><br>"
    print l,"<br>"
    print mw_dict[l], "<br>"
    print "%.2f <br>" % labeling[l]
    print "</td>"
print "<td>"
# print flags for potentially problematic wells
if (foundNotLabeled == 0):
    print "!!! NO Unlabeled !!! <br>"
if (unidentified == 1):
    print "!!! Unidentified >0.3 !!! <br>"
if (doubleLabel == 1):
    print "!!! Double label !!! <br>"
if (adductConfusion != 0):
    print "!!! "+ adductConfusion +"is ~ mw of comp <br>"
if (baseChange > 1):
    print "!!! BaseChange "+ str(baseChange)
if (badWell > 4):
    print "!!! BAD WELL"
print "</td>"

return labeling

##### MAIN #####

#read in the Expected MW of the compounds
mw = {}
fmw = open ('ExpMW.txt');
mwlines = fmw.readlines()
fmw.close()

for l in mwlines:
    l.strip()
    mw[l.split()[0]]=float(l.split()[1])

#read in the Plate Map
map = {}
fmap = open ('PlateMap.txt');
maplines = fmap.readlines()
fmap.close()

for l in maplines:
    l.strip()
    map[l.split()[0]]=l.split()[1:]

#read in text report file
f = open(sys.argv[1])
lines = f.readlines()
f.close()

nlines=len(lines)

```

```

i=0

#skip the initial plate char graphic
while ("Sample" not in lines[i]):
    i = i+1
i = i+1

#store sample and well numbers
samples = []
while (lines[i][0].isdigit()):
    samples.append(lines[i].rstrip().split()[1])
    i = i+1

#identify correct peaks (assuming they're 100%
correctPeaks = []
while (len(correctPeaks) < len(samples)):
    if (len(lines[i].split())>12):
        if (lines[i].split()[12]=="100.000"):
            correctPeaks.append(lines[i].split()[0])
            i = i+1
        else:
            i = i+1
    else:
        i = i+1

#store spectra
peakSpectra = [{}]*(len(samples))
currSample = 0
currPeak = 0
while (i<nlines-1):
    if (len(lines[i].split())>0) and (lines[i].split()
[0]=="Sample"):
        currSample = int(lines[i].split()[1])
        i = i+1

    if (len(lines[i].split())>0) and (lines[i].split()[0]=="Peak"):
        currPeak = lines[i].split()[1]
        i = i+1

    if correctPeaks[currSample-1]==currPeak:
        i = i+1
        tmpdict = {}
        while (lines[i][0].isdigit()):
            tmpdict[float(lines[i].split()
[0])]=float(lines[i].split()[1])
            i = i+1
        peakSpectra[currSample-1] = tmpdict
        i = i+1

#choose top10 spectra for each sample
topSpectra = [{}]*(len(samples))
for l in peakSpectra:
    sorted_l = sorted(l.items(), key=operator.itemgetter(1),
reverse=True)

```

```

        #print len(sorted_l)
        topSpectra.append(sorted_l[0:10])

#generate HTML code for nice presentation of results
print "<html>"
print "<head></head>"
print "<body>"
print "<table>"

#assign labeling % for each well
unassigned=[]
overall_label={}
label = {}
i=0
while (i<len(samples)):
    print "<tr>"
    #get well number
    char=samples[i].split(':')[1].split(',')[0]
    num=samples[i].split(':')[1].split(',')[1]
    well="%s%02d" %(char,int(num))
    print "<td>",well,"</td> <td>"
    ##### !!!! Assumes last argument is reference well with just the
protein #####
    label =
    assign_sample(unassigned,well,mw,map,topSpectra[i],topSpectra[0])
    for l in label:
        overall_label[l]=label[l]
    print "</td></tr>"
    i = i+1

#close html
print "</table>"
print "</body>"
print "</html>"

#make labeling results file
f = open('overall.labeling', 'w')
for l in overall_label:
    if not 'protein' in l:
        f.write("%s %.2f\n" %(l,overall_label[l]))
f.close()

i=0
while i < len(unassigned):
    print unassigned[i][0],unassigned[i][1]
    i = i+1

#print images according to labeling %
f = open('sorted.html', 'w')
f.write("<html><head></head><body><table><tr>")
perline=0
sorted_overall = sorted(overall_label.items(),
key=operator.itemgetter(1), reverse=True)

```

```
for s in sorted_overall:
    perline = perline+1

    f.write("<td>")
    f.write("<img src='../pngs/'+s[0]+'.png'
width=145><br>" + s[0] + "<br>" + str(s[1]) + "<br>")
    f.write("</td>")

    if perline==7:
        f.write("</tr><tr>")
        perline=0

f.write("</tr></table></body></html>")
f.close()
```