# Constans

from urllib.request import urlretrieve

from pyopenms.Constants import \*

import pyopenms

from pyopenms import ElementDB, EmpiricalFormula, CoarseIsotopePatternGenerator, FineIsotopePatternGenerator, ResidueDB, \

ModificationsDB, RibonucleotideDB, AASequence, Residue, FASTAEntry, FASTAFile

import math

from matplotlib import pyplot as plt

help(pyopenms.Constants)

print("Avogadro's number :", pyopenms.Constants.AVOGADRO)

######################################

#Elements

DataBase = ElementDB()

DataBase.hasElement("H")

DataBase.hasElement("He")

hydrogen = DataBase.getElement("H")

print(hydrogen.getName())

print(hydrogen.getSymbol())

print(hydrogen.getMonoWeight())

print(hydrogen.getAverageWeight())

#################################

heleim = DataBase.getElement("He")

print(heleim.getName())

print(heleim.getSymbol())

print(heleim.getMonoWeight())

print(heleim.getAverageWeight())

isotopes = heleim.getIsotopeDistribution()

print("One mole of hydrogen weighs equals =", 2 \* hydrogen.getAverageWeight(), "grams")

print("One mole of 16O2 weighs equals=", 2 \* hydrogen.getMonoWeight(), "grams")

##########################################

#Isotops

hydrogn\_isoDist = {"mass": [], "abundance": []}

heliem\_isoDist = {"mass": [], "abundance": []}

hydrogen = DataBase.getElement("H")

isotopes = hydrogen.getIsotopeDistribution()

for iso in isotopes.getContainer():

print("hydrogen isotope:", iso.getMZ(), "has abundance", iso.getIntensity() \* 100, "%")

hydrogn\_isoDist["mass"].append(iso.getMZ())

hydrogn\_isoDist["abundance"].append((iso.getIntensity() \* 100))

heleim = DataBase.getElement("He")

isotopes = heleim.getIsotopeDistribution()

for iso in isotopes.getContainer():

print("heleim isotope", iso.getMZ(), "has abundance", iso.getIntensity() \* 100, "%")

heliem\_isoDist["mass"].append(iso.getMZ())

heliem\_isoDist["abundance"].append((iso.getIntensity() \* 100))

def adjustText(x1, y1, x2, y2):

if y1 > y2:

plt.annotate('%0.3f' % (y2), xy=(x2, y2), xytext=(x2 + 0.5, y2 + 9),

textcoords='data',

arrowprops=dict(arrowstyle="->", color='r', lw=0.5),

horizontalalignment='right', verticalalignment='top')

else:

plt.annotate('%0.3f' % (y1), xy=(x1, y1), xytext=(x1 + 0.5, y1 + 9),

textcoords='data',

arrowprops=dict(arrowstyle="->", color='r', lw=0.5),

horizontalalignment='right', verticalalignment='top')

#######################################

def plotDistribution(distribution):

n = len(distribution["mass"])

for i in range(0, n):

plt.vlines(x=distribution["mass"][i], ymin=0, ymax=distribution["abundance"][i])

if int(distribution["mass"][i - 1]) == int(distribution["mass"][i]) \

and i != 0:

adjustText(distribution["mass"][i - 1], distribution["abundance"][i - 1],

distribution["mass"][i], distribution["abundance"][i])

else:

plt.text(x=distribution["mass"][i],

y=(distribution["abundance"][i] + 2),

s='%0.3f' % (distribution["abundance"][i]), va='center',

ha='center')

plt.ylim([0, 110])

plt.xticks(range(math.ceil(distribution["mass"][0]) - 2,

math.ceil(distribution["mass"][-1]) + 2))

#######################################

plt.figure(figsize=(10, 7))

plt.subplot(1, 2, 1)

plt.title("Isotopic distribution of Hydrogen")

plotDistribution(hydrogn\_isoDist)

plt.xlabel("Atomic mass (u)")

plt.ylabel("Relative abundance (%)")

#######################################

plt.subplot(1, 2, 2)

plt.title("Isotopic distribution of Heleim")

plotDistribution(heliem\_isoDist)

plt.xlabel("Atomic mass (u)")

plt.ylabel("Relative abundance (%)")

plt.show()

#########################################

#Mass defect

isotopes = DataBase.getElement("C").getIsotopeDistribution().getContainer()

carbon\_isotope\_difference = isotopes[1].getMZ() - isotopes[0].getMZ()

isotopes = DataBase.getElement("N").getIsotopeDistribution().getContainer()

nitrogen\_isotope\_difference = isotopes[1].getMZ() - isotopes[0].getMZ()

print("Mass difference between 12C and 13C:", carbon\_isotope\_difference)

print("Mass difference between 14N and N15:", nitrogen\_isotope\_difference)

print("Relative deviation:", 100 \* (carbon\_isotope\_difference - nitrogen\_isotope\_difference) / carbon\_isotope\_difference, "%")

################################

helium = ElementDB().getElement("He")

isotopes = helium.getIsotopeDistribution()

mass\_sum = 2 \* PROTON\_MASS\_U + 2 \* ELECTRON\_MASS\_U + 2 \* NEUTRON\_MASS\_U

helium4 = isotopes.getContainer()[1].getMZ()

print("Sum of masses of 2 protons, neutrons and electrons:", mass\_sum)

print("Mass of He4:", helium4)

print("Difference between the two masses:", 100 \* (mass\_sum - helium4) / mass\_sum, "%")

#Molcular formula

methanol = EmpiricalFormula("CH3OH")

water = EmpiricalFormula("H2O")

ethanol = EmpiricalFormula("CH2") + methanol

print("Ethanol chemical formula:", ethanol.toString())

print("Ethanol composition:", ethanol.getElementalComposition())

print("Ethanol has", ethanol.getElementalComposition()[b"H"], "hydrogen atoms")

# Isotopic Distributions

methanol\_isoDist = {"mass": [], "abundance": []}

ethanol\_isoDist = {"mass": [], "abundance": []}

##############################

print("Coarse Isotope Distribution:")

isotopes = ethanol.getIsotopeDistribution(CoarseIsotopePatternGenerator(4))

prob\_sum = sum([iso.getIntensity() for iso in isotopes.getContainer()])

print("This covers", prob\_sum, "probability")

for iso in isotopes.getContainer():

print("Isotope", iso.getMZ(), "has abundance", iso.getIntensity() \* 100, "%")

methanol\_isoDist["mass"].append(iso.getMZ())

methanol\_isoDist["abundance"].append((iso.getIntensity() \* 100))

#####################################

print("Fine Isotope Distribution:")

isotopes = ethanol.getIsotopeDistribution(FineIsotopePatternGenerator(1e-3))

prob\_sum = sum([iso.getIntensity() for iso in isotopes.getContainer()])

print("This covers", prob\_sum, "probability")

for iso in isotopes.getContainer():

print("Isotope", iso.getMZ(), "has abundance", iso.getIntensity() \* 100, "%")

ethanol\_isoDist["mass"].append(iso.getMZ())

ethanol\_isoDist["abundance"].append((iso.getIntensity() \* 100))

#######################################

plt.figure(figsize=(10, 7))

plt.subplot(1, 2, 1)

plt.title("Isotopic distribution of methanol")

plotDistribution(methanol\_isoDist)

plt.xlabel("Atomic mass (u)")

plt.ylabel("Relative abundance (%)")

plt.subplot(1, 2, 2)

plt.title("Isotopic distribution of ethanol")

plotDistribution(ethanol\_isoDist)

plt.xlabel("Atomic mass (u)")

plt.ylabel("Relative abundance (%)")

plt.savefig("methanol\_ethanol\_isoDistribution.png")

#######################################

print("Fine Isotope Distribution:")

isotopes = ethanol.getIsotopeDistribution(FineIsotopePatternGenerator(1e-6))

prob\_sum = sum([iso.getIntensity() for iso in isotopes.getContainer()])

print("This covers", prob\_sum, "probability")

for iso in isotopes.getContainer():

print("Isotope", iso.getMZ(), "has abundance", iso.getIntensity() \* 100, "%")

#########################################

isotopes = ethanol.getIsotopeDistribution(CoarseIsotopePatternGenerator(5, True))

for iso in isotopes.getContainer():

print("Isotope", iso.getMZ(), "has abundance", iso.getIntensity() \* 100, "%")

#########################################

#Amino Acid Modifications

Am = ResidueDB().getResidue("Lysine")

print(Am.getName())

print(Am.getThreeLetterCode())

print(Am.getOneLetterCode())

print(Am.getAverageWeight())

print(Am.getMonoWeight())

print(Am.getPka())

print(Am.getFormula().toString())

#############################

ox = ModificationsDB().getModification("Oxidation")

print(ox.getUniModAccession())

print(ox.getUniModRecordId())

print(ox.getDiffMonoMass())

print(ox.getId())

print(ox.getFullId())

print(ox.getFullName())

print(ox.getDiffFormula())

################################

isotopes = ox.getDiffFormula().getIsotopeDistribution(CoarseIsotopePatternGenerator(5))

for iso in isotopes.getContainer():

print(iso.getMZ(), ":", iso.getIntensity())

##########################################################

# Ribonucleotides

uridine = RibonucleotideDB().getRibonucleotide(b"U")

print(uridine.getName())

print(uridine.getCode())

print(uridine.getAvgMass())

print(uridine.getMonoMass())

print(uridine.getFormula().toString())

print(uridine.isModified())

methyladenosine = RibonucleotideDB().getRibonucleotide(b"m1A")

print(methyladenosine.getName())

print(methyladenosine.isModified())

###################################

#Amino Acid Sequences

seq\_3 = AASequence.fromString("DFPIANGER")

prefix = seq\_3.getPrefix(4)

suffix = seq\_3.getSuffix(5)

concat = seq\_3 + seq\_3

print("Sequence:", seq\_3)

print("Prefix:", prefix)

print("Suffix:", suffix)

print("Concatenated:", concat)

mfull = seq\_3.getMonoWeight()

mprecursor = seq\_3.getMonoWeight(Residue.ResidueType.Full, 2)

mz = seq\_3.getMonoWeight(Residue.ResidueType.Full, 2) / 2.0

print("Monoisotopic mass of peptide [M] is", mfull)

print("Monoisotopic mass of peptide precursor [M+2H]2+ is", mprecursor)

print("Monoisotopic m/z of [M+2H]2+ is", mz)

#######################################

print("The peptide", str(seq\_3), "consists of the following amino acids:")

for aa in seq\_3:

print(aa.getName(), ":", aa.getMonoWeight())

########################################

seq\_2 = AASequence.fromString("C[143]PKCK(Label:13C(6)15N(2))CR")

if seq\_2.hasNTerminalModification():

print("N-Term Modification: ", seq\_2.getNTerminalModification().getFullId())

if seq\_2.hasCTerminalModification():

print("C-Term Modification: ", seq\_2.getCTerminalModification().getFullId())

for aa in seq\_2:

if (aa.isModified()):

print(aa.getName(), ":", aa.getMonoWeight(), ":", aa.getModificationName())

else:

print(aa.getName(), ":", aa.getMonoWeight())

#######################################

#Molecular formula

seq\_formula = seq\_3.getFormula()

print("Peptide", seq\_3, "has molecular formula", seq\_formula)

########################################

#Isotope patterns

coarse\_isotopes = seq\_formula.getIsotopeDistribution(CoarseIsotopePatternGenerator(6))

for iso in coarse\_isotopes.getContainer():

print("Isotope", iso.getMZ(), "has abundance", iso.getIntensity() \* 100, "%")

fine\_isotopes = seq\_formula.getIsotopeDistribution(FineIsotopePatternGenerator(0.01))

for iso in fine\_isotopes.getContainer():

print("Isotope", iso.getMZ(), "has abundance", iso.getIntensity() \* 100, "%")

##########################################

def plotIsotopeDistribution(isotope\_distribution, title="Isotope distribution"):

plt.title(title)

distribution = {"mass": [], "abundance": []}

for iso\_2 in isotope\_distribution.getContainer():

distribution["mass"].append(iso\_2.getMZ())

distribution["abundance"].append(iso\_2.getIntensity() \* 100)

bars = plt.bar(distribution["mass"], distribution["abundance"], width=0.01,

snap=False)

plt.ylim([0, 110])

plt.xticks(range(math.ceil(distribution["mass"][0]) - 2,

math.ceil(distribution["mass"][-1]) + 2))

plt.xlabel("Atomic mass (u)")

plt.ylabel("Relative abundance (%)")

plt.figure(figsize=(10, 7))

plt.subplot(1, 2, 1)

plotIsotopeDistribution(coarse\_isotopes, "Isotope distribution - coarse")

plt.subplot(1, 2, 2)

plotIsotopeDistribution(fine\_isotopes, "Isotope distribution - fine structure")

plt.show()

##########################################

#Fragment ions

suffix = seq\_3.getSuffix(3)

print("=" \* 35)

print("y3 ion sequence:", suffix)

y3\_formula = suffix.getFormula(Residue.ResidueType.YIon, 2)

suffix.getMonoWeight(Residue.ResidueType.YIon, 2) / 2.0

suffix.getMonoWeight(Residue.ResidueType.XIon, 2) / 2.0

suffix.getMonoWeight(Residue.ResidueType.BIon, 2) / 2.0

print("y3 mz:", suffix.getMonoWeight(Residue.ResidueType.YIon, 2) / 2.0)

print("y3 molecular formula:", y3\_formula)

##########################################

#Modified Sequences

seq\_3 = AASequence.fromString("PEPTIDESEKUEM(Oxidation)CER")

print(seq\_3.toUnmodifiedString())

print(seq\_3.toString())

print(seq\_3.toUniModString())

print(seq\_3.toBracketString())

print(seq\_3.toBracketString(False))

print(AASequence.fromString("DFPIAM(UniMod:35)GER"))

print(AASequence.fromString("DFPIAM[+16]GER"))

print(AASequence.fromString("DFPIAM[+15.99]GER"))

print(AASequence.fromString("DFPIAM[147]GER"))

print(AASequence.fromString("DFPIAM[147.035405]GER"))

#######################################

s = AASequence.fromString(".(Dimethyl)DFPIAMGER.")

print(s, s.hasNTerminalModification())

s = AASequence.fromString(".DFPIAMGER.(Label:18O(2))")

print(s, s.hasCTerminalModification())

s = AASequence.fromString(".DFPIAMGER(Phospho).")

print(s, s.hasCTerminalModification())

#######################################

#Proteins and FASTA files

bsa = FASTAEntry()

bsa.sequence = "MKWVTFISLLLLFSSAYSRGVFRRDTHKSEIAHRFKDLGE"

bsa.description = "BSA Bovine Albumin (partial sequence)"

bsa.identifier = "BSA"

alb = FASTAEntry()

alb.sequence = "MKWVTFISLLFLFSSAYSRGVFRRDAHKSEVAHRFKDLGE"

alb.description = "ALB Human Albumin (partial sequence)"

alb.identifier = "ALB"

entries = [bsa, alb]

f = FASTAFile()

f.store("example.fasta", entries)

entries = []

f = FASTAFile()

f.load("example.fasta", entries)

print(len(entries))

for e in entries:

print(e.identifier, e.sequence)

#Task1

seq = AASequence.fromString("VAKA")

seq\_formula = seq.getFormula()

vakaTotalMZ=0

coarse\_isotopes = seq\_formula.getIsotopeDistribution( CoarseIsotopePatternGenerator(6) )

for iso in coarse\_isotopes.getContainer():

print ("Isotope", iso.getMZ(),)

vakaTotalMZ+=iso.getMZ()

print(vakaTotalMZ)

#######################################

v = ResidueDB().getResidue("V")

a = ResidueDB().getResidue("A")

k = ResidueDB().getResidue("K")

l=[v,a,k,a]

subVakaMZ=0;

for i in l:

vf=EmpiricalFormula(v.getFormula().toString()).getIsotopeDistribution(CoarseIsotopePatternGenerator(5))

for iso in vf.getContainer():

subVakaMZ+=iso.getMZ()

print(subVakaMZ)

#Task3

from pyopenms import \*

from urllib.request import urlretrieve

gh = "https://raw.githubusercontent.com/OpenMS/pyopenms-extra/master"

urlretrieve (gh + "/src/data/P02769.fasta", "123.fasta")

dig = ProteaseDigestion()

dig.getEnzymeName() # Trypsin

bsa = "".join([l.strip() for l in open("123.fasta").readlines()[1:]])

bsa = AASequence.fromString(bsa)

# create all digestion products

result = []

dig.digest(bsa, result)

print(result[4].toString())

len(result)

#####################################

dig = ProteaseDigestion()

dig.setEnzyme('Lys-C')

bsa = "".join([l.strip() for l in fh.readlines()[1:]])

bsa = AASequence.fromString(bsa)

result = []

dig.digest(bsa, result)

print(result[4].toString())

len(result)