Python

# merge data

import pandas as pd

import xlrd

w1 = xlrd.open\_workbook('2021MCM\_ProblemC\_ Images\_by\_GlobalID.xlsx')

w2 = xlrd.open\_workbook('2021MCMProblemC\_DataSet.xlsx')

sheet\_1 = w1.sheet\_by\_index(0)

sheet\_2 = w2.sheet\_by\_index(0)

data\_row = []

for row in range(sheet\_1.nrows):

data\_row.append(sheet\_1.row\_values(row))

data\_row2 = []

for row in range(sheet\_2.nrows):

data\_row2.append(sheet\_2.row\_values(row))

from pandas.core.frame import DataFrame

df1 = DataFrame(data\_row)

df2 = DataFrame(data\_row2)

mg = pd.merge(df2,df1, left\_on = 0, right\_on = 1, how='left')

mg.drop\_duplicates()

print(mg)

mg.to\_excel('data\_merged.xlsx')

# images classification

import os

import shutil

import re

import pandas as pd

import xlrd

path = "D:\\SCHOOL\\INCLASS\\20-21spring\\OTHERS\\2021America\\C\\2021MCM\_ProblemC\_Files\\2021MCM\_ProblemC\_Files"

file\_list = os.listdir(path)

print(file\_list)

w1 = xlrd.open\_workbook('data\_merged3.xlsx')

sheet\_1 = w1.sheet\_by\_index(0)

LabStatus = sheet\_1.col\_values(3)

"def col\_values(self, colx, start\_rowx=0, end\_rowx=None):"

GlobalID = sheet\_1.col\_values(0)

"def col\_values(self, colx, start\_rowx=0, end\_rowx=None):"

FileName = sheet\_1.col\_values(8)

"def col\_values(self, colx, start\_rowx=0, end\_rowx=None):"

m = -1

for i in LabStatus:

if i == 'Unprocessed':

m = m + 1

print(i)

print(FileName[m])

if FileName[m] != '':

if os.path.exists(os.path.join("C:\\Users\\HW\\Desktop\\2021MCM\_ProblemC\_Files", FileName[m])):

print('hello')

shutil.move(os.path.join("C:\\Users\\HW\\Desktop\\2021MCM\_ProblemC\_Files", FileName[m]),"C:\\Users\\HW\Desktop\\2")

else:

m = m + 1

pass

# Location

import pandas as pd

import xlrd

from pandas.core.frame import DataFrame

import math

import numpy

w = xlrd.open\_workbook('data\_merged4.xlsx')

sheet\_1 = w.sheet\_by\_index(0)

positive\_location\_19 = [[48.98099,48.971949,49.025831,48.993892,49.149394],[-122.688503,-122.700941,-122.810653,-122.702242,-123.943134]]

Latitude = sheet\_1.col\_values(6)

"def col\_values(self, colx, start\_rowx=0, end\_rowx=None):"

Latitude = Latitude[1:]

Longitude = sheet\_1.col\_values(7)

"def col\_values(self, colx, start\_rowx=0, end\_rowx=None):"

Longitude = Longitude[1:]

m = 0

location = []

e = []

f = []

for i in Longitude:

b = 1000

d = 1000

for j in range(5):

a = abs(Longitude[m] - positive\_location\_19[1][j]) \* 111.7

if a < b:

b = a

else:

pass

for k in range(5):

c = abs(math.cos(Latitude[m] / 180 \* math.pi) \* 111.7 \* (abs(Latitude[m] - positive\_location\_19[0][k])))

if c < d:

d = c

#print(b)

e.append(b)

f.append(d)

#print(d)

m = m + 1

dist = []

dist2 = []

for i in range(len(e)):

distnum = e[i] \* e[i] + f[i] \* f[i]

print(distnum)

if distnum > (8 \* 1.61 \* 8 \* 1.61):

distjudge = 0

else:

distjudge = 1

dist.append(distjudge)

dist2.append(distnum)

#print(dist)

distD = DataFrame(dist)

distD.to\_excel('dist6.xlsx')

# time

import pandas as pd

import xlrd

from pandas.core.frame import DataFrame

import math

w = xlrd.open\_workbook('data\_merged4.xlsx')

sheet\_1 = w.sheet\_by\_index(0)

DetectionTime = sheet\_1.col\_values(1)

"def col\_values(self, colx, start\_rowx=0, end\_rowx=None):"

m = 0

DetTime = []

for i in DetectionTime:

#print(str(DetectionTime[m])[:-2])

a = str(DetectionTime[m])[:-2]

m = m + 1

DetTime.append(a)

m = 0

mat = []

for i in DetTime:

if DetTime[m].isdigit():

print(DetectionTime[m] % 365)

mat.append(DetectionTime[m] % 365)

m = m + 1

else:

mat.append(400)

m = m + 1

mat

m = 0

time = []

for i in mat:

if mat[m] >= 122:

if mat[m] <= 304:

#print('a')

print(1 / (1 + 99 \* math.exp(-0.051 \* (mat[m]-122))))

time.append(1 / (1 + 99 \* math.exp(-0.051 \* (mat[m]-122))))

m = m + 1

elif mat[m] <= 365:

#print('b')

print(1 / math.exp(0.037 \* (mat[m]-305)))

time.append(1 / math.exp(0.037 \* (mat[m]-305)))

m = m + 1

else:

#print('d')

print(0)

time.append('0')

m = m + 1

elif mat[m] >= 43:

#print('c')

m = m + 1

print(0.01)

time.append('0.01')

else:

#print('b')

print(1 / math.exp(0.037 \* (mat[m]+81)))

time.append(1 / math.exp(0.037 \* (mat[m]+81)))

m = m + 1

timeD = DataFrame(time)

timeD.to\_excel('mat6.xlsx')

R

# suitability

library(readxl)

data = read.table("3\_InputDataForMigClim.csv",header=TRUE, sep=",")

data\_merged3 = read\_excel("data\_merged3.xlsx")

data\_merged3 = data\_merged3[c("Longitude", "Latitude")]

suit\_matrix = list()

for (i in (1:5618)){

distance = ((data$x - data\_merged3$Longitude[i]) ^ 2 + (data$y - data\_merged3$Latitude[i]) ^ 2) ^ (1/2)

distance\_bind = cbind(data,distance)

distance

data\_near = order(distance\_bind[,"distance"], decreasing = FALSE)

data\_near

suitability\_sum = 0

for (x in 1:10){suitability\_sum = suitability\_sum + distance\_bind[data\_near,][x,4]

suitability <- suitability\_sum / 10

}

print(suitability)

suit\_matrix = rbind(suit\_matrix,suitability)

}

write.csv(suit\_matrix,file = "try.csv")

# descriptive statistics and glm

data = read.csv("data\_merged7.csv")

data\_pn= data[data$Lab.Status == "Positive ID" | data$Lab.Status == "Negative ID",

c("Lab.Status","suitability","time\_normalization","Location","Image")]

data\_n = data[data$Lab.Status == "Negative ID",

c("Lab.Status","suitability","time\_normalization","Location","Image")]

data\_p = data[data$Lab.Status == "Positive ID",

c("Lab.Status","suitability","time\_normalization","Location","Image")]

data\_pn <- data\_pn %>%

mutate(Lab.Status = if\_else(Lab.Status == "Positive ID", 1, 0))

summary(data\_pn$suitability)

summary(data$time\_normalization)

data\_pn= data[data$Lab.Status == "Positive ID" | data$Lab.Status == "Negative ID",

c("Lab.Status","suitability","time\_normalization","Location","Image")]

data\_pn <- data\_pn %>%

mutate(Location = as.factor(Location),

Lab.Status = as.factor(Lab.Status)

) %>%

mutate\_if(is.character, as.factor) %>%

dplyr::select(Lab.Status,Location, everything())

ggplot(data, aes(x=Lab.Status, fill=Lab.Status)) +

geom\_bar() +

xlab("Lab Status") +

ylab("Count")

ggplot(data, aes(x=suitability)) +

geom\_histogram() +

#scale\_fill\_manual(values = "red") +

scale\_fill\_brewer(palette = "Paired") +

xlab("Suitability") +

ylab("Count")

ggplot(data, aes(x=time\_normalization)) +

#geom\_histogram(fill = "red",color = "red") +

geom\_histogram() +

stat\_bin(bins = 10) +

xlab("Time") +

ylab("Count")

p0 <- ggplot(data\_pn, aes(Lab.Status, fill = Location))+

geom\_bar(position = "fill")+ylab("Percentage")

p0

set.seed(399973)

data\_p$Lab.Status = 1

m = sample(1:3237,17)

data\_n\_sample = data.frame()

for (i in (0:17)){data\_n\_sample = rbind(data\_n\_sample,data\_n[m[i],])}

data\_n\_sample$Lab.Status = 0

ourdata = rbind(data\_n\_sample,data\_p)

fit\_full = glm(Lab.Status~., family = binomial(link = "logit"), data = ourdata,

control=list(maxit=1000))

as\_flextable(fit\_full)

fit\_red = glm(Lab.Status~suitability+Location+Image, family = binomial(link = "logit"),

data = ourdata, control=list(maxit=1000))

as\_flextable(fit\_red)

vif\_red = as.data.frame(vif(fit\_red))

vif\_red

anova = anova(fit\_red, fit\_full,test = "Chisq")

flextable(data = anova)

coef = as.data.frame(exp(coef(fit\_red)))

names(coef) = "Coefficient"

coef

as.data.frame(exp(confint(fit\_red)))

fit.od = glm(Lab.Status~., family = quasibinomial(link = "logit"), data = ourdata,

control=list(maxit=100))

pchisq(summary(fit.od)$dispersion \* fit\_red$df.residual,fit\_red$df.residual, lower = F)

true = as.factor(ourdata$Lab.Status)

pred = as.numeric(predict(fit\_red,type = "response", ourdata)>0.5)

pred = as.factor(pred)

confusion\_matrix = confusionMatrix(data = pred, reference = true)

confusion\_matrix