```
#####################################################################
# Step 3: Optimize SNF Supply Logistics
# Optimizes factory locations, capacities, and supply chains
# Optimizer minimizes costs while treating all SAM and MAM cases
# Runs the parameter study by changing variable costs
#####################################################################

from pulp import *
import math
import json
import numpy as np
import re
from sys import exit
import matplotlib.cm as cm

try:
    wddata='/Users/lilllianpetersen/iiasa/data/supply_chain/'
    wdfigs='/Users/lilllianpetersen/iiasa/figs/'
    wdvars='/Users/lilllianpetersen/iiasa/saved_vars/'
    f=open(wddata+'trading_across_borders2017.csv','r')
except:
    wddata='C:/Users/garyk/Documents/code/riskAssessmentFromPovertyEstimations/supply_chain/data/'
    wdfigs='C:/Users/garyk/Documents/code/riskAssessmentFromPovertyEstimations/supply_chain/figs/'
    wdvars='C:/Users/garyk/Documents/code/riskAssessmentFromPovertyEstimations/supply_chain/vars/'
# 'AllIntl_opti',

countryCostedTariff = np.load(wdvars+'tariff_by_country.npy')

bigloop=True

if(bigloop):

    optiLevel = ['AllarOpti','LocalOpti','AllIntl']
    loopvar = ['shipcost', 'impexp','strtup','truckfactor', 'tariff']
    mins= np.array([0.2,0.2,0.5,0.2, 0])
    factor = np.array([0.2,0.2,0.5,0.2, 0.2])
    maxs = np.array([2.01,4.01,9.51,4.01, 2.6])
else:
    import matplotlib.pyplot as plt
    optiLevel=['AllarOpti']
    # optiLevel=['AllIntl_trf','AllarOpti','LocalOpti','AllIntl_opti_trf', 'LocalOpti_trf','AllarOpti_trf
    loopvar=['shipcost']
    mins=np.array([1])
    factor=np.array([1])
    maxs=np.array([1.1])

for k in range(len(optiLevel)):
        # for x in range(len(countrycosted)):
        #     if(tmp[0]==countrycosted[x] or tmp[0][2:]==countrycosted[x]):
        #         scaleaverage[x]=tmp[1][:-2]
        #     else:
        #         print tmp[0]

    #     transportcostArray=np.zeros(shape=(33,43))
    #     f=open(wddata+'travel_time/INTLcapitaldistanceArray.csv','r')
    #     i=-1
    #     for line in f:
    #         i+=1
    #         tmp=line.split(',')
    #         for j in range(len(tmp)):
    #             avg=(scaleaverage[i]+Fscaleaverage[j])/2.
    #             transportcostArray[i,j]=float(tmp[j])*avg*(1/1000.)
    #
    #     # import and export costs
    #     importExportCosts=np.zeros(shape=(transportcostArray.shape))
```

```
#     for x in range(len(countrycosted)):
#        exportCost=-9999
#        fx=open(wddata+'trading_across_borders2017.csv','r')
#        if(countrycosted[x][0:2]=='I_'):
#            xCountry=countrycosted[x][2:]
#            for line in fx:
#       tmp=line.split(',')
#       if tmp[0]==xCountry:
#           exportCost=float(tmp[4])+float(tmp[6])+3330.
#           break
#                for y in range(len(subsaharancountry)):
#       importCost=-9999
#       yCountry=subsaharancountry[y]
#       if xCountry==yCountry:
#         continue
#
#       fy=open(wddata+'trading_across_borders2017.csv','r')
#       for line in fy:
#          tmp=line.split(',')
#          if tmp[0]==yCountry:
#             importCost=float(tmp[8])+float(tmp[10])
#             break
#
#       importExportCosts[x,y]=importCost+exportCost
#        else:
#            xCountry=countrycosted[x]
#            for line in fx:
#                tmp=line.split(',')
#                if tmp[0]==xCountry:
#                    exportCost=float(tmp[4])+float(tmp[6])
#                    break
#        # print exportCost,xCountry
#
#            for y in range(len(subsaharancountry)):
#           importCost=-9999
#           yCountry=subsaharancountry[y]
#           if xCountry==yCountry:
#          continue
#
#         fy=open(wddata+'trading_across_borders2017.csv','r')
#         for line in fy:
#         tmp=line.split(',')
#         if tmp[0]==yCountry:
#         importCost=float(tmp[8])+float(tmp[10])
#         break
#
#         importExportCosts[x,y]=importCost+exportCost
for z in range(len(loopvar)):
    mShip=False
    mImpExp=False
    mStrt=False
    mTruck=False
    mTariff=False
    if(loopvar[z]=='shipcost'):
        mShip=True
    elif(loopvar[z]=='impexp'):
        mImpExp=True
    elif(loopvar[z]=='strtup'):
        mStrt=True
    elif(loopvar[z]=='truckfactor'):
        mTruck=True
    elif(loopvar[z]=='tariff'):
        mTariff=True
    for s in np.arange(mins[z],maxs[z],factor[z]):
        print 's=',s
```

```
print 'optiLevel=',optiLevel[k],k
print 'loopvar=',loopvar[z] ,z
##capitalonly
subsaharancountry=[]
subsaharancapital=[]
indexedwasting=np.zeros(shape=43)
indexedSAM=np.zeros(shape=43)
indexedstunting=np.zeros(shape=43)
indexedMAM=np.zeros(shape=43)
f=open(wddata+'population/CAPITALVERSIONcasenumbers.csv','r')
i=-1
for line in f:
    i+=1
    tmp=line.split(',')
    subsaharancountry.append(tmp[0])
    indexedwasting[i]=float(tmp[1])
    indexedSAM[i]=float(tmp[2])*150*11.66
    indexedMAM[i]=float(tmp[3])*50*(365/75)
    indexedstunting[i]=float(tmp[4])
    subsaharancapital.append(tmp[5][:-2])
if (optiLevel[k]=='AllarOpti'):
    ### cost per country
    countrycosted=[]
    rutfprice=[]
    rusfprice=[]
    scplusprice=[]
    capitalcosted=[]
    f=open(wddata+'foodstuffs/pricesCorrected_intl_opti.csv')
    code=np.zeros(shape=(247),dtype=int)
    i=-1
    for line in f:
        i+=1
        tmp=line.split(',')
        countrycosted.append(tmp[0])
        rutfprice.append(tmp[1])
        rusfprice.append(tmp[2])
        scplusprice.append(tmp[3])
        capitalcosted.append(tmp[4][:-2])
elif (optiLevel[k]=='AllarOpti_trf'):
    ### cost per country
    countrycosted=[]
    rutfprice=[]
    rusfprice=[]
    scplusprice=[]
    capitalcosted=[]
    f=open(wddata+'foodstuffs/pricesCorrected_intl_opti_trf.csv')
    code=np.zeros(shape=(247),dtype=int)
    i=-1
    for line in f:
        i+=1
        tmp=line.split(',')
        countrycosted.append(tmp[0])
        rutfprice.append(tmp[1])
        rusfprice.append(tmp[2])
        scplusprice.append(tmp[3])
        capitalcosted.append(tmp[4][:-2])
elif (optiLevel[k]=='LocalOpti'):
    ### cost per country
    countrycosted=[]
    rutfprice=[]
    rusfprice=[]
    scplusprice=[]
    capitalcosted=[]
    f=open(wddata+'foodstuffs/pricesCorrected_intl.csv')
    code=np.zeros(shape=(247),dtype=int)
```

```
    i=-1
    for line in f:
        i+=1
        tmp=line.split(',')
        countrycosted.append(tmp[0])
        rutfprice.append(tmp[1])
        rusfprice.append(tmp[2])
        scplusprice.append(tmp[3])
        capitalcosted.append(tmp[4][:-2])
elif (optiLevel[k]=='LocalOpti_trf'):
    ### cost per country
    countrycosted=[]
    rutfprice=[]
    rusfprice=[]
    scplusprice=[]
    capitalcosted=[]
    f=open(wddata+'foodstuffs/pricesCorrected_intl_trf.csv')
    code=np.zeros(shape=(247),dtype=int)
    i=-1
    for line in f:
        i+=1
        tmp=line.split(',')
        countrycosted.append(tmp[0])
        rutfprice.append(tmp[1])
        rusfprice.append(tmp[2])
        scplusprice.append(tmp[3])
        capitalcosted.append(tmp[4][:-2])
elif (optiLevel[k]=='AllIntl'):
    ### cost per country
    countrycosted=[]
    rutfprice=[]
    rusfprice=[]
    scplusprice=[]
    capitalcosted=[]
    f=open(wddata+'foodstuffs/pricesCorrected_AllIntl.csv')
    code=np.zeros(shape=(247),dtype=int)
    i=-1
    for line in f:
        i+=1
        tmp=line.split(',')
        countrycosted.append(tmp[0])
        rutfprice.append(tmp[1])
        rusfprice.append(tmp[2])
        scplusprice.append(tmp[3])
        capitalcosted.append(tmp[4][:-2])
elif (optiLevel[k]=='AllIntl_trf'):
    ### cost per country
    countrycosted=[]
    rutfprice=[]
    rusfprice=[]
    scplusprice=[]
    capitalcosted=[]
    f=open(wddata+'foodstuffs/pricesCorrected_AllIntl_trf.csv')
    code=np.zeros(shape=(247),dtype=int)
    i=-1
    for line in f:
        i+=1
        tmp=line.split(',')
        countrycosted.append(tmp[0])
        rutfprice.append(tmp[1])
        rusfprice.append(tmp[2])
        scplusprice.append(tmp[3])
        capitalcosted.append(tmp[4][:-2])
elif (optiLevel[k]=='AllIntl_opti'):
    ### cost per country
```

```
    countrycosted=[]
    rutfprice=[]
    rusfprice=[]
    scplusprice=[]
    capitalcosted=[]
    f=open(wddata+'foodstuffs/pricesCorrected_AllIntl_opti.csv')
    code=np.zeros(shape=(247),dtype=int)
    i=-1
    for line in f:
           i+=1
           tmp=line.split(',')
           countrycosted.append(tmp[0])
           rutfprice.append(tmp[1])
           rusfprice.append(tmp[2])
           scplusprice.append(tmp[3])
           capitalcosted.append(tmp[4][:-2])
elif (optiLevel[k]=='AllIntl_opti_trf'):
    ### cost per country
    countrycosted=[]
    rutfprice=[]
    rusfprice=[]
    scplusprice=[]
    capitalcosted=[]
    f=open(wddata+'foodstuffs/pricesCorrected_AllIntl_opti_trf.csv')
    code=np.zeros(shape=(247),dtype=int)
    i=-1
    for line in f:
           i+=1
           tmp=line.split(',')
           countrycosted.append(tmp[0])
           rutfprice.append(tmp[1])
           rusfprice.append(tmp[2])
           scplusprice.append(tmp[3])
           capitalcosted.append(tmp[4][:-2])
countrycosted[6]="Congo (Republic of the)"
countrycosted[7]="Cote d'Ivoire"

facility=countrycosted
location=subsaharancountry

indexedrutf=np.zeros(shape=43)
indexedrusf=np.zeros(shape=43)
indexedscp=np.zeros(shape=43)
conversionlist=[]
convertarray=np.zeros(shape=43)
for i in range(len(countrycosted)):
    for j in range(len(subsaharancountry)):
        if countrycosted[i]==subsaharancountry[j]:
            conversionlist.append(j)
            convertarray[j]=i
            indexedrutf[j]=rutfprice[i]
            indexedrusf[j]=rusfprice[i]
            indexedscp[j]=scplusprice[i][:-1]

# #for all
# Fscaleaverage=np.zeros(shape=43)
# f=open(wddata+'travel_time/averagetkmcost.csv','r')
# i=-1
# for line in f:
#     tmp=line.split(',')
#     i+=1
#     Fscaleaverage[i]=tmp[1][:-1]
#
# #for facilities
# scaleaverage=np.zeros(shape=33)
```

```
# f=open(wddata+'travel_time/averagetkmcost.csv','r')
# i=-1
# for line in f:
#     tmp=line.split(',')
#     i+=1
#     countrycosted=np.array(countrycosted)
#     c=np.where(tmp[0]==countrycosted)[0]
#     c2=np.where(['I_'+tmp[0]]==countrycosted)[0]
#
#     if len(c)!=0:
#         c=c[0]
#         scaleaverage[c]=tmp[1][:-1]
#     if len(c2)!=0:
#         c2=c2[0]
#         scaleaverage[c2]=tmp[1][:-1]
mShipV=1
mImpExpV=1
mStrtV=1
mTruckV=1
mTariffV=1
if(mShip==True):
    mShipV=s
elif(mImpExp==True):
    mImpExpV=s
elif(mStrt==True):
    mStrtV=s
elif(mTruck==True):
    mTruckV=s
elif(mTariff==True):
    mTariffV=s
transportcostArray = np.load(wddata+'travel_time/totalTruckingCost.npy')
transportcostArray = mTruckV*transportcostArray/1000
# transportcostArray=np.zeros(shape=(33,43))
# f=open(wddata+'travel_time/INTLcapitaldistanceArray.csv','r')
# i=-1
# for line in f:
#     i+=1
#     tmp=line.split(',')
#     for j in range(len(tmp)):
#         avg=(scaleaverage[i]+Fscaleaverage[j])/2.
#         transportcostArray[i,j]=float(tmp[j])*avg*(1/1000.)*mTruckV

# import and export costs
importExportCosts=np.zeros(shape=(transportcostArray.shape))
for x in range(len(countrycosted)):
    exportCost=-9999
    fx=open(wddata+'trading_across_borders2017.csv','r')
    if(countrycosted[x][0:2]=='I_'):
        xCountry=countrycosted[x][2:]
        for line in fx:
            tmp=line.split(',')
            if tmp[0]==xCountry:
                exportCost=mImpExpV*(float(tmp[4])+float(tmp[6]))+mShipV*2550.
                break
        for y in range(len(subsaharancountry)):
            importCost=-9999
            yCountry=subsaharancountry[y]
            if xCountry==yCountry:
                continue

            fy=open(wddata+'trading_across_borders2017.csv','r')
            for line in fy:
                tmp=line.split(',')
                if tmp[0]==yCountry:
                    importCost=mImpExpV*(float(tmp[8])+float(tmp[10]))
```

```
                    break
            importExportCosts[x,y]=importCost+exportCost
    else:
        xCountry=countrycosted[x]
        for line in fx:
            tmp=line.split(',')
            if tmp[0]==xCountry:
                exportCost=mImpExpV*(float(tmp[4])+float(tmp[6]))
                break

        for y in range(len(subsaharancountry)):
            importCost=-9999
            yCountry=subsaharancountry[y]
            if xCountry==yCountry:
                continue
            fy=open(wddata+'trading_across_borders2017.csv','r')
            for line in fy:
                tmp=line.split(',')
                if tmp[0]==yCountry:
                    importCost=mImpExpV*(float(tmp[8])+float(tmp[10]))
                    break
            importExportCosts[x,y]=importCost+exportCost

#cost dabber RUTF ###############################################################################
rutfcostarray=np.zeros(shape=(33,43))
for i in range(len(countrycosted)):
    for j in range(len(indexedSAM)):
# sums ingredient and transport cost, converts to $/100g delivered
        rutfcostarray[i,j]=rutfprice[i]

rutfdictionary={}
### array to dict
for i in range(len(countrycosted)):
    rutfdictionary[countrycosted[i]]={}
    for j in range(len(subsaharancountry)):
        if(countrycosted[i][:2]=='I_'):
            rutfdictionary[countrycosted[i]][subsaharancountry[j]]=rutfcostarray[i,j]
        else:
            if(optiLevel[k]=='AllIntl'):
                rutfdictionary[countrycosted[i]][subsaharancountry[j]]=rutfcostarray[i,j]+rut
            else:
                rutfdictionary[countrycosted[i]][subsaharancountry[j]]=rutfcostarray[i,j]+rut

with open(wddata+'optiarrays/rutfdictionary.json', 'w') as fp:
    json.dump(rutfdictionary, fp, sort_keys=True)

np.savetxt(wddata + "foodstuffs/rutfcostarray.csv", rutfcostarray, delimiter=",")

#cost dabber MAM ################################################################################
SCPuse=np.zeros(shape=(33,43))
mamcostarray=np.zeros(shape=(33,43))
for i in range(len(countrycosted)):
    for j in range(len(indexedSAM)):
            if(float(scplusprice[i])*2<float(rusfprice[i])):
                for j in range(len(indexedSAM)):
                    mamcostarray[i,j]=2*float(scplusprice[i])
                    SCPuse[i,j]=True
            else:
                for j in range(len(indexedSAM)):
                    mamcostarray[i,j]=float(rusfprice[i])

np.savetxt(wddata + "foodstuffs/mamcostarray.csv", mamcostarray, delimiter=",")

mamdictionary={}
### array to dict
```

```
for i in range(len(countrycosted)):
    mamdictionary[countrycosted[i]]={}
    for j in range(len(subsaharancountry)):
        if(countrycosted[i][:2]=='I_'):
            mamdictionary[countrycosted[i]][subsaharancountry[j]]=mamcostarray[i,j]
        else:
            if(optiLevel[k]=='AllIntl'):
                mamdictionary[countrycosted[i]][subsaharancountry[j]]=mamcostarray[i,j]+mamcos
            else:
                mamdictionary[countrycosted[i]][subsaharancountry[j]]=mamcostarray[i,j]+mamcos

with open(wddata+'optiarrays/mamdictionary.json', 'w') as fp:
    json.dump(mamdictionary, fp, sort_keys=True)

# transport cost dabber
samtransportcostarray=np.zeros(shape=(33,43))
for i in range(len(countrycosted)):
    for j in range(len(indexedSAM)):
        samtransportcostarray[i,j]=(109.5/1000000.)*transportcostArray[i,j]+(109.5/1000000.)*:

mamtransportcostarray=np.zeros(shape=(33,43))
for i in range(len(countrycosted)):
    for j in range(len(indexedSAM)):
        if(SCPuse[i,j]):
            mamtransportcostarray[i,j]=(209.5/1000000.)*transportcostArray[i,j]+(209.5/100000(
        else:
            mamtransportcostarray[i,j]=(109.5/1000000.)*transportcostArray[i,j]+(109.5/100000(

samtransportcostdictionary={}
### array to dict
for i in range(len(countrycosted)):
    samtransportcostdictionary[countrycosted[i]]={}
    for j in range(len(subsaharancountry)):
        samtransportcostdictionary[countrycosted[i]][subsaharancountry[j]]=samtransportcostarı

mamtransportcostdictionary={}
### array to dict
for i in range(len(countrycosted)):
    mamtransportcostdictionary[countrycosted[i]]={}
    for j in range(len(subsaharancountry)):
        mamtransportcostdictionary[countrycosted[i]][subsaharancountry[j]]=mamtransportcostarı

#OPTIMIZER

#fixed costs for facility
# startupcost= 1000000.0
strt=1000000.*0.2*mStrtV
startupcost = dict(zip(facility, [strt, strt, strt, strt, strt, strt, strt, strt, strt, strt,

upgrd=20000.*0.2*mStrtV
upgradecost= dict(zip(facility, [upgrd, upgrd, upgrd, upgrd, upgrd, upgrd, upgrd, upgrd, upgr(

#machinery costs for production, small machinery
m1c=6000.*0.2*mStrtV
costM1 = dict(zip(facility, [m1c, m1c, m1c, m1c, m1c, m1c, m1c, m1c, m1c, m1c, m1c, m1c, m1c,
#machinery costs for production, large machinery

m2c=10000.*0.2*mStrtV
costM2 = dict(zip(facility, [m2c, m2c, m2c, m2c, m2c, m2c, m2c, m2c, m2c, m2c, m2c, m2c, m2c,
#fixed capacity per piece of machinery, small machinery
Capacity1 = 780000
#fixed capacity per piece of machinery, large machinery
Capacity2 = (780000*2.0)
#demand by location
rutfdemandarray = np.genfromtxt(wddata+'optiarrays/SAMdemand.csv', delimiter=',')
```

```
DemandRUTF = dict(zip(location, rutfdemandarray))

mamdemandarray = np.genfromtxt(wddata+'optiarrays/MAMdemand.csv', delimiter=',')
DemandMAM = dict(zip(location, mamdemandarray))

# cost to location
# with open(wddata+'optiarrays/rutfdictionary.json', 'r') as fp:
#     data = json.load(fp)
CostRUTF = rutfdictionary

# with open(wddata+'optiarrays/mamdictionary.json', 'r') as fp:
#     data = json.load(fp)
CostMAM = mamdictionary

MAMCostTransport = mamtransportcostdictionary

SAMCostTransport = samtransportcostdictionary

QuantityRUTF = LpVariable.dicts ('Supply of RUTF %s%s',(facility, location),cat = 'Continuous

QuantityMAM = LpVariable.dicts ('Supply of MAM Treatment %s%s',(facility, location),cat = 'Cor

#number of small machines
Machine1 = LpVariable.dicts('Machine 1 %s', facility,
                    lowBound = 0,
                    cat='Integer')
#number of large machines
Machine2 = LpVariable.dicts('Machine 2 %s', facility,
                    lowBound = 0,
                    cat='Integer')
# factory open? Y/N
Open = LpVariable.dicts('Factory Status %s', facility,
                    cat='Binary')
#factory size upgrades
Factorysize = LpVariable.dicts('Factory Size %s', facility,
                    lowBound = 0,
                    cat='Integer')

prob = LpProblem('Fixed Charge', LpMinimize)
tmp1 = sum(costM1[i] * Machine1[i] for i in facility)
tmp2 = sum(costM2[i] * Machine2[i] for i in facility)
tmp3 = sum(startupcost[i] * Open[i] for i in facility)
tmp4 = sum(upgradecost[i] * Factorysize[i] for i in facility)
tmp5 = sum(sum((CostRUTF[i][j] * QuantityRUTF[i][j]) + (SAMCostTransport[i][j] * QuantityRUTF[
tmp6 = sum(sum((CostMAM[i][j] * QuantityMAM[i][j]) + (MAMCostTransport[i][j] * QuantityMAM[i][
prob+=tmp1+tmp2+tmp3+tmp4+tmp5+tmp6

#must be less than small machinery
for i in facility:
    prob += sum(QuantityRUTF[i][j]+QuantityMAM[i][j] for j in location) <= (Capacity1*Machine
#must be less than maximum amount a factory can produce (this ensures fixed cost is used)
for i in facility:
    prob += sum(QuantityRUTF[i][j]+QuantityMAM[i][j] for j in location) <= 100000000000000*Ope
for i in facility:
    prob += sum(Machine2[i]+Machine1[i]) <= 10*Factorysize[i]
for j in location:
    prob += sum(QuantityRUTF[i][j] for i in facility) >= DemandRUTF[j]
for j in location:
    prob += sum(QuantityMAM[i][j] for i in facility) >= DemandMAM[j]

prob.solve()
# print("Status:")
# print(LpStatus[prob.status])
# print("Objective:")
# print s
```

```
# print z
print(value(prob.objective))
# for v in prob.variables():
#     if(v.varValue>0):
#         print (v.name, "=", v.varValue)

varsdict = {}
for v in prob.variables():
    if(v.varValue>0):
        varsdict[v.name] = v.varValue

with open(wddata+'optiarrays/temp_results.json', 'w') as fp:
    json.dump(varsdict, fp, sort_keys=True)


for i in range(len(subsaharancountry)):
    subsaharancountry[i]=subsaharancountry[i].replace(' ','_')

for i in range(len(countrycosted)):
    countrycosted[i]=countrycosted[i].replace(' ','_')

for i in range(len(capitalcosted)):
    capitalcosted[i]=capitalcosted[i].replace(' ','_')

cost=0
factorynum=0
sourcenum=0
averageshipments=0
countries=[]
cost=value(prob.objective)
sizes=[]
mamsup=[]
samsup=[]
for v in prob.variables():
    if (v.varValue>0):
        if (v.name[0:10]=="Factory_St"):
            # print v.name[15:]
            sourcenum+=1
            countries.append(v.name[15:])
            if('I_' not in v.name):
                factorynum+=1
        if (v.name[0:9]=="Machine_2"):
            sizes.append(value(v))
        if (v.name[0:10]=="Supply_of_" and not ('I_' in v.name)):
            averageshipments+=1

if(factorynum!=0):
    averageshipments=(averageshipments/2.)/factorynum
else:
    averageshipments=0

rutfsupplyarray=np.zeros(shape=(33,43))

for i in countrycosted:
    j=0
    for v in prob.variables():
        if ("Supply_of_RUTF" in v.name and v.name[15:int(15+len(i))]==i):
            a=np.where(np.array(countrycosted)==i)[0][0]
            rutfsupplyarray[a,j]=v.varValue
            j+=1

rutftotaled=np.sum(rutfsupplyarray,axis=1)

rusfsupplyarray=np.zeros(shape=(33,43))
for i in countrycosted:
```

```
    j=0
    for v in prob.variables():
        if ("Supply_of_MAM" in v.name and v.name[24:int(24+len(i))]==i):
            a=np.where(np.array(countrycosted)==i)[0][0]
            rusfsupplyarray[a,j]=v.varValue
            j+=1

rusftotaled=np.sum(rusfsupplyarray,axis=1)

ingredientcosttotalpercent=np.sum(rutfsupplyarray*rutfcostarray+rusfsupplyarray*mamcostarray),

MAMtransportcostarray=np.zeros(shape=(33,43))
RUTFtransportcostarray=np.zeros(shape=(33,43))
for i in range(len(countrycosted)):
    for j in range(len(indexedSAM)):
        RUTFtransportcostarray[i,j]=(109.5/1000000.)*transportcostArray[i,j]+(109.5/1000000.)*
        if(SCPuse[i,j]==True):
            MAMtransportcostarray[i,j]=(209.5/1000000.)*transportcostArray[i,j]+(209.5/100000(
        else:
            MAMtransportcostarray[i,j]=(109.5/1000000.)*transportcostArray[i,j]+(109.5/100000(


transportpercent = np.sum(MAMtransportcostarray*rusfsupplyarray+RUTFtransportcostarray*rutfsup

if(bigloop):
    ##recipetype_factorchanged
    if not os.path.exists(wddata+'/results/'+str(optiLevel[k])+'_'+str(loopvar[z])+'/'):
        os.makedirs(wddata+'/results/'+str(optiLevel[k])+'_'+str(loopvar[z])+'/')
    f = open(wddata+'/results/'+str(optiLevel[k])+'_'+str(loopvar[z])+'/' +str(optiLevel[k])+
    f.write('cost'+','+str(cost)+'\n')
    f.write('num_factories'+','+str(factorynum)+'\n')
    f.write('percent_transport'+','+str(transportpercent)+'\n')
    f.write('percent_ingredient'+','+str(ingredientcosttotalpercent)+'\n')
    f.write('average_shipments_per_local+factory'+','+str(averageshipments)+'\n')
    for i in range(len(countrycosted)):
        if(rutftotaled[i]!=0 or rusftotaled[i]!=0):
            f.write(str(countrycosted[i])+','+str(rutftotaled[i])+','+str(rusftotaled[i])+'\n
    f.close()
else:
    if not os.path.exists(wddata+'/results/special/'+str(optiLevel[k])+'_'+str(loopvar[z])+'/
        os.makedirs(wddata+'/results/special/'+str(optiLevel[k])+'_'+str(loopvar[z])+'/')
    f = open(wddata+'/results/special/'+str(optiLevel[k])+'_'+str(loopvar[z])+'/' +str(optiLev
    f.write('cost'+','+str(cost)+'\n')
    f.write('num_factories'+','+str(factorynum)+'\n')
    f.write('percent_transport'+','+str(transportpercent)+'\n')
    f.write('percent_ingredient'+','+str(ingredientcosttotalpercent)+'\n')
    f.write('average_shipments_per_local+factory'+','+str(averageshipments)+'\n')
    for i in range(len(countrycosted)):
        if(rutftotaled[i]!=0 or rusftotaled[i]!=0):
            f.write(str(countrycosted[i])+','+str(rutftotaled[i])+','+str(rusftotaled[i])+'\n
    f.close()
Rcountrycosted=[]
lngth=len(np.where(rutftotaled + rusftotaled>0)[0])
Rrutfsupplyarray=np.zeros(shape=(lngth,43))
Rrusfsupplyarray=np.zeros(shape=(lngth,43))
o=0
for i in range(len(countrycosted)):
    if(rutftotaled[i]!=0 or rusftotaled[i]!=0):
        Rcountrycosted.append(countrycosted[i])
        Rrutfsupplyarray[o,:]=rutfsupplyarray[i,:]
        Rrusfsupplyarray[o,:]=rusfsupplyarray[i,:]
        o+=1

Rcountrycosted=np.array(Rcountrycosted)
countryToiF = {}
```

```
iToCountryF = {}
for i in range(len(Rcountrycosted)):
    countryToiF[Rcountrycosted[i]] = i
    iToCountryF[i]=Rcountrycosted[i]

countryToiC = {}
iToCountryC = {}
for i in range(len(subsaharancountry)):
    countryToiC[subsaharancountry[i]] = i
    iToCountryC[i]=subsaharancountry[i]

lonsF=np.zeros(shape=(lngth))
f = open(wddata+'boundaries/countryLats.csv')
for line in f:
    line=line[:-1]
    tmp=line.split(',')
    if np.amax(tmp[3]==np.array(Rcountrycosted))==0:
        continue
    i=countryToiF[tmp[3]]
    lonsF[i]=float(tmp[2])


lonsC=np.zeros(shape=(43))
f = open(wddata+'boundaries/countryLats_noI.csv')
for line in f:
    line=line[:-1]
    tmp=line.split(',')
    if np.amax(tmp[3]==np.array(subsaharancountry))==0:
        continue
    i=countryToiC[tmp[3]]
    lonsC[i]=float(tmp[2])


iSortedF = np.argsort(lonsF)
iSortedC = np.argsort(lonsC)

subsaharancountry =  np.array(subsaharancountry)

RNcountrycosted = Rcountrycosted
RNsubsaharancountry = subsaharancountry
RNrutfsupplyarray = Rrutfsupplyarray
RNrusfsupplyarray = Rrusfsupplyarray

Rcountrycosted = Rcountrycosted[iSortedF]
Rsubsaharancountry = subsaharancountry[iSortedC]

Rrutfsupplyarray=Rrutfsupplyarray[iSortedF]
Rrutfsupplyarray=Rrutfsupplyarray[:,iSortedC]

Rrusfsupplyarray=Rrusfsupplyarray[iSortedF]
Rrusfsupplyarray=Rrusfsupplyarray[:,iSortedC]

if not os.path.exists(wddata+'/results/example/'+str(optiLevel[k])+'/'):
    os.makedirs(wddata+'/results/example/'+str(optiLevel[k])+'/')

np.save(wddata+'/results/example/'+str(optiLevel[k])+"/RNcountry", RNcountrycosted)
np.save(wddata+'/results/example/'+str(optiLevel[k])+"/RNsubsaharancountry", RNsubsaharancount
np.save(wddata+'/results/example/'+str(optiLevel[k])+ "/RNrutfarray", RNrutfsupplyarray)
np.save(wddata+'/results/example/'+str(optiLevel[k])+ "/RNrusfarray", RNrusfsupplyarray)

np.save(wddata+'/results/example/'+str(optiLevel[k])+"/Rcountry", Rcountrycosted)
np.save(wddata+'/results/example/'+str(optiLevel[k])+"/Rsubsaharancountry", Rsubsaharancountry
np.save(wddata+'/results/example/'+str(optiLevel[k])+ "/Rrutfarray", Rrutfsupplyarray)
np.save(wddata+'/results/example/'+str(optiLevel[k])+ "/Rrusfarray", Rrusfsupplyarray)
```

```
np.save(wddata+'/boundaries/countrycosted',countrycosted)
np.save(wddata+'/boundaries/subsaharancountry',subsaharancountry)

rusfarray = np.load(wddata+'/results/example/'+str(optiLevel[k])+'/Rrusfarray.npy')
rutfarray = np.load(wddata+'/results/example/'+str(optiLevel[k])+'/Rrutfarray.npy')
Rcountry = np.load(wddata+'/results/example/'+str(optiLevel[k])+'/Rcountry.npy')
Rcountry[Rcountry=='Congo']='DRC'
Rcountry[Rcountry=='Congo_(Republic_of_the)']='Congo'
Rcountry[Rcountry=='Cote_d\'Ivoire']='Ivory Coast'

Rcountry2=[]
for i in range(len(Rcountry)):
    country=Rcountry[i]
    if country[:2]=='I_':
        if country=="I_Cote_d'Ivoire":
            country='I_Ivory_Coast'
        Rcountry2.append('Intl ('+country[2:]+')')
    else:
        Rcountry2.append(country)
Rcountry2=np.array(Rcountry2)
for i in range(len(Rcountry2)):
    Rcountry2[i]=Rcountry2[i].replace('_',' ')

# plt.clf()
# fig = plt.figure(figsize=(13, 8))
# plt.imshow(rutfarray,cmap=cm.terrain_r,vmax=2.5e8)
# plt.colorbar()
# plt.xticks(np.arange(43), Rsubsaharancountry, rotation='vertical')
# plt.yticks(np.arange(len(Rcountry2)), Rcountry2)
# plt.title('Imports and Exports RUTF (Packets, Ordered by Lon)')
# plt.ylabel('Suppliers')
# plt.xlabel('Recipients')
# plt.savefig(wddata+'/results/example/'+str(optiLevel[k])+'/RUTFimpexp_array.pdf')
#
# plt.clf()
# fig = plt.figure(figsize=(13, 8))
# plt.imshow(rusfarray,cmap=cm.terrain_r,vmax=2.5e8)
# plt.colorbar()
# plt.xticks(np.arange(43), Rsubsaharancountry, rotation='vertical')
# plt.yticks(np.arange(len(Rcountry2)), Rcountry2)
# plt.title('Imports and Exports RUSF (Packets, Ordered by Lon)')
# plt.ylabel('Suppliers')
# plt.xlabel('Recipients')
# plt.savefig(wddata+'/results/example/'+str(optiLevel[k])+'/RUSFimpexp_array.pdf')
#
# rutfarray=rutfarray+rusfarray
#
# plt.clf()
# fig = plt.figure(figsize=(13, 8))
# plt.imshow(rutfarray,cmap=cm.terrain_r,vmax=2.5e8)
# plt.colorbar()
# plt.xticks(np.arange(43), Rsubsaharancountry, rotation='vertical')
# plt.yticks(np.arange(len(Rcountry2)), Rcountry2)
# plt.title('Total Treatments Delivered (Packets, Ordered by Lon)')
# plt.ylabel('Suppliers')
# plt.xlabel('Recipients')
# plt.savefig(wddata+'/results/example/'+str(optiLevel[k])+'/TOTALimpexp_array.pdf')
# rutfarray=0

#     plt.clf()
#     plt.imshow(Rrutfsupplyarray,cmap=cm.terrain_r)
#     plt.colorbar()
#     plt.title('RUTF Imports and Exports ordered by Lon')
#     plt.savefig(wdfigs+'rutfimpexp.png')
#
```

```
    #     plt.clf()
    #     plt.imshow(Rrusfsupplyarray,cmap=cm.terrain_r)
    #     plt.colorbar()
    #     plt.title('RUSF treatment Imports and Exports ordered by Lon')
    #     plt.savefig(wdfigs+'rusfimpexp.png')
    #
    # if v.name[0,5]="

# resultsarray=np.zeros(shape=(86,2))
# i=-1
# for v in varsdict():
#     if(v.varValue>0 and v.name[:4] == 'Supp'):
#     i+=1
#     # resultsarray[i,0]=v.name
#     resultsarray[i,1]=v.value
#     # if(v.name[10:14]=="RUTF"):
#     #     resultsarray[i,0]= 'RUTF'
#     # else:
#     #     resultsarray[i,0]= 'RUSF'
#     # tmp = re.findall('[A-Z][^A-Z]*', v.name[15:])
#     # resultsarray[i,1]=float(tmp[0])
#     # resultsarray[i,2]=v.varValue
#
# np.savetxt(wddata + "optiarrays/results/results.csv", resultsarray, delimiter=",")
```