**README FILE**

This document has 3 parts:

- The SOFTWARE needed to run the python code,

- The FILES needed to run the python and cplex code,

- The python CODE itself.

**FIRST PART: The SOFTWARE needed to run the python code.**

- IBM ILOG Optimization Studio should be installed (<https://www.ibm.com/support/pages/downloading-ibm-ilog-cplex-optimization-studio-v1290>).

After installation, create an empty project and add the files mentioned below to its working directory (to solve the linear optimization problem).

- MS Office Excel should be installed (to read and write .xlsx files).

- Python should be installed. Preferably install Anaconda ([Anaconda | Individual Edition](https://www.anaconda.com/products/individual)) and install Spyder Python IDE with python (to loop the ILOG Optimizer).

*When using Anaconda or similar package managers, please note that any missing packages to run the code should be installed through the package manager. If you encounter such a problem, look up how to install the specific package to the package manager and not the IDE.*

**SECOND PART: The FILES needed to run the python code.**

Inside the python compiler directory (for example C:\Users\user\_name\.spyder-py3\):

- loop\_for\_cplex.py.

Inside the cplex compiler directory (for example C:\Users\user\_name\opl\project\_name\):

- data.xlsx,

- part.dat,

- part.mod.

**THIRD PART: The python CODE itself.** You can inspect the python code with Microsoft Visual Studio Code or any other python IDE (such as Spyder).

**WARNING:** Close Excel and ILOG Optimization Studio before running the code!

**WARNING 2:** Make sure that ALL pathing is correct – adjust each highlighted line in the code accordingly! (You can do this by using the search & replace function in any compiler by pressing Ctrl + F and searching for “C:\\Users\\Reha\\opl\\thesis\\”. Replace each occurrence with “C:\\Users\\user\_name\\opl\\project\_name\\”.)

import pandas as pd

import numpy as np

import time

from subprocess import call

from openpyxl import Workbook

t = time.time()

# close Excel and ILOG Studio before running the code

data = pd.read\_excel('C:\\Users\\rehat\\opl\\project1\\data.xlsx', sheet\_name="python")

df2 = Workbook()

df2.save(filename='C:\\Users\\rehat\\opl\\project1\\money\_allocation.xlsx') # prepare the results file for the money allocation task

df2bis = Workbook()

df2bis.save(filename='C:\\Users\\rehat\\opl\\project1\\time\_allocation.xlsx') # prepare the results file for the time allocation task

df3 = [] # to calculate ratios and store types for money allocation task

df3bis = [] # to calculate ratios and store types for time allocation task

char = "C" # to increment where we write the results at each s

c = ord(char[0])

'''

step 0 - sampling from data

- set nb\_draws=XXX for the number of draws

- set sample\_size=XXX for the fraction of the original data we want in each draw

NOTE: Running the whle script from the beginning will overwrite the results file

'''

nb\_draws= 200 # set the number of sampling

sample\_size= 0.6 # set the sample size as a fraction of the original data

for s in range(nb\_draws): # Set s to the desired draw and run from this line to avoid overwriting results !

looptime = time.time() - t

print('Loop ' + str(s+1) + ' started, elapsed time: ' + str(looptime))

df = data.sort\_values("ID")

df = df.sample(frac=sample\_size) # add random\_state to set seed and to compare both tasks

df.to\_excel('C:\\Users\\rehat\\opl\\project1\\data\_for\_python.xlsx', index=False)

'''

1st step - no categories (finding tau\_hat) for MONEY ALLOCATION TASK

'''

i=1 # counter for excel cells (to properly store output in results file)

with open("C:\\Users\\rehat\\opl\\project1\\part.dat",'w') as f:

f.write('NR\_Goods=2;\n'+'SheetConnection comm("data\_for\_python.xlsx");\n')

f.write('NR\_Observations= ' + str(len(df)) + ';\n')

f.write("""AllP from SheetRead(comm,"'Sheet1'!E2:F"""+ str(len(df)+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!C2:D"""+ str(len(df)+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!G2:G"""+ str(len(df)+1)+"""");\n""")

f.write('SheetConnection comm2("money\_allocation.xlsx");\n')

f.write("""NR\_Types to SheetWrite(comm2,"'Sheet'!"""+str(chr(c))+""""""+str(i)+""":"""+str(chr(c))+""""""+str(i)+"""");\n""")

f.write("""NR\_Observations to SheetWrite(comm2,"'Sheet'!D"""+str(i)+""":D"""+str(i)+"""");\n""")

f.write("""runtype = "Money\_Allocation";\n""")

f.write("""runtype to SheetWrite(comm2,"'Sheet'!A"""+str(i)+""":A"""+str(i)+"""");\n""")

f.write('runtype2 = " ";\n')

f.write("""runtype2 to SheetWrite(comm2,"'Sheet'!B"""+str(i)+""":B"""+str(i)+"""");\n""")

call(["oplrun.exe", "C:\\Users\\rehat\\opl\\project1\\part.mod", "C:\\Users\\rehat\\opl\\project1\\part.dat"])

i=i+1 # increment by 1 at the end of each run

'''

2nd step - loop over k and l (finding tau\_k)

'''

for k in df[['Gender','Age','Education','Marital Status','Employment']]:

df = df.sort\_values(k)

df.to\_excel('C:\\Users\\rehat\\opl\\project1\\data\_for\_python.xlsx', index=False)

d = df.groupby([k]).size().reset\_index(name='Count')

for l in range(len(d)): # counter for each state in an observable char

with open("C:\\Users\\rehat\\opl\\project1\\part.dat",'w') as f:

f.write('NR\_Goods=2;\n'+'SheetConnection comm("data\_for\_python.xlsx");\n')

f.write('NR\_Observations= ' + str(d['Count'][l]) + ';\n')

if l==0:

f.write("""AllP from SheetRead(comm,"'Sheet1'!E2:F"""+ str(d['Count'][l]+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!C2:D"""+ str(d['Count'][l]+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!G2:G"""+ str(d['Count'][l]+1)+"""");\n""")

n=d['Count'][l]

elif l>0:

f.write("""AllP from SheetRead(comm,"'Sheet1'!E"""+str(n+2)+""":F"""+ str(n+d['Count'][l]+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!C"""+str(n+2)+""":D"""+ str(n+d['Count'][l]+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!G"""+str(n+2)+""":G"""+ str(n+d['Count'][l]+1)+"""");\n""")

n=n+d['Count'][l]

f.write('SheetConnection comm2("money\_allocation.xlsx");\n')

f.write("""NR\_Types to SheetWrite(comm2,"'Sheet'!"""+str(chr(c))+""""""+str(i)+""":"""+str(chr(c))+""""""+str(i)+"""");\n""")

f.write("""NR\_Observations to SheetWrite(comm2,"'Sheet'!D"""+str(i)+""":D"""+str(i)+"""");\n""")

f.write('runtype = "'+str(k)+'";\n')

f.write("""runtype to SheetWrite(comm2,"'Sheet'!A"""+str(i)+""":A"""+str(i)+"""");\n""")

f.write('runtype2 = "'+str(d[k][l])+'";\n')

f.write("""runtype2 to SheetWrite(comm2,"'Sheet'!B"""+str(i)+""":B"""+str(i)+"""");\n""")

call(["oplrun.exe", "C:\\Users\\rehat\\opl\\project1\\part.mod", "C:\\Users\\rehat\\opl\\project1\\part.dat"])

i=i+1 # increment by 1 at the end of each run

n=0 # reset to 0 at the end of k

l=0 # reset to 0 at the end of k

'''

3rd step - loop over k and l given j (find tau\_k,j)

'''

for j in df[['Gender','Age','Education','Marital Status','Employment']]:

for k in df[['Gender','Age','Education','Marital Status','Employment']]:

# there should be an elegant way to do this but let's do it manually since it's just 5 choose 2

if j == 'Age' and k == 'Gender':

continue

if j == 'Education' and (k == 'Gender' or k == 'Age'):

continue

if j == 'Marital Status' and (k == 'Gender' or k == 'Age' or k == 'Education'):

continue

if j == 'Employment' and (k == 'Gender' or k == 'Age' or k == 'Education' or k == 'Marital Status'):

continue

if j == k:

continue

df = df.sort\_values([j,k])

df.to\_excel('C:\\Users\\rehat\\opl\\project1\\data\_for\_python.xlsx', index=False)

d = df.groupby([j,k]).size().reset\_index(name='Count')

for l in range(len(d)): # counter for each state in an observable char

with open("C:\\Users\\rehat\\opl\\project1\\part.dat",'w') as f:

f.write('NR\_Goods=2;\n'+'SheetConnection comm("data\_for\_python.xlsx");\n')

f.write('NR\_Observations= ' + str(d['Count'][l]) + ';\n')

if l==0:

f.write("""AllP from SheetRead(comm,"'Sheet1'!E2:F"""+ str(d['Count'][l]+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!C2:D"""+ str(d['Count'][l]+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!G2:G"""+ str(d['Count'][l]+1)+"""");\n""")

n=d['Count'][l]

elif l>0:

f.write("""AllP from SheetRead(comm,"'Sheet1'!E"""+str(n+2)+""":F"""+ str(n+d['Count'][l]+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!C"""+str(n+2)+""":D"""+ str(n+d['Count'][l]+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!G"""+str(n+2)+""":G"""+ str(n+d['Count'][l]+1)+"""");\n""")

n=n+d['Count'][l]

f.write('SheetConnection comm2("money\_allocation.xlsx");\n')

f.write("""NR\_Types to SheetWrite(comm2,"'Sheet'!"""+str(chr(c))+""""""+str(i)+""":"""+str(chr(c))+""""""+str(i)+"""");\n""")

f.write("""NR\_Observations to SheetWrite(comm2,"'Sheet'!D"""+str(i)+""":D"""+str(i)+"""");\n""")

f.write('runtype2 = "'+str(d[j][l])+' and ' +str(d[k][l])+ '";\n')

f.write("""runtype2 to SheetWrite(comm2,"'Sheet'!B"""+str(i)+""":B"""+str(i)+"""");\n""")

f.write('runtype = "'+str(j)+' and ' +str(k)+'";\n')

f.write("""runtype to SheetWrite(comm2,"'Sheet'!A"""+str(i)+""":A"""+str(i)+"""");\n""")

call(["oplrun.exe", "C:\\Users\\rehat\\opl\\project1\\part.mod", "C:\\Users\\rehat\\opl\\project1\\part.dat"])

i=i+1 # increment by 1 at the end of each run

n=0 # reset to 0 at the end of k

l=0 # reset to 0 at the end of k

df3.append(pd.read\_excel('C:\\Users\\rehat\\opl\\project1\\money\_allocation.xlsx', header=None))

'''

1bis - no categories (finding tau\_hat) for TIME ALLOCATION TASK

'''

i=1 # counter for excel cells (to properly store output in results file)

with open("C:\\Users\\rehat\\opl\\project1\\part.dat",'w') as f:

f.write('NR\_Goods=2;\n'+'SheetConnection comm("data\_for\_python.xlsx");\n')

f.write('NR\_Observations= ' + str(len(df)) + ';\n')

f.write("""AllP from SheetRead(comm,"'Sheet1'!K2:L"""+ str(len(df)+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!I2:J"""+ str(len(df)+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!M2:M"""+ str(len(df)+1)+"""");\n""")

f.write('SheetConnection comm2("time\_allocation.xlsx");\n')

f.write("""NR\_Types to SheetWrite(comm2,"'Sheet'!"""+str(chr(c))+""""""+str(i)+""":"""+str(chr(c))+""""""+str(i)+"""");\n""")

f.write("""NR\_Observations to SheetWrite(comm2,"'Sheet'!D"""+str(i)+""":D"""+str(i)+"""");\n""")

f.write("""runtype = "Time\_Allocation";\n""")

f.write("""runtype to SheetWrite(comm2,"'Sheet'!A"""+str(i)+""":A"""+str(i)+"""");\n""")

f.write('runtype2 = " ";\n')

f.write("""runtype2 to SheetWrite(comm2,"'Sheet'!B"""+str(i)+""":B"""+str(i)+"""");\n""")

call(["oplrun.exe", "C:\\Users\\rehat\\opl\\project1\\part.mod", "C:\\Users\\rehat\\opl\\project1\\part.dat"])

i=i+1 # increment by 1 at the end of each run

'''

2bis - loop over k and l (finding tau\_k)

'''

for k in df[['Gender','Age','Education','Marital Status','Employment']]:

df = df.sort\_values(k)

df.to\_excel('C:\\Users\\rehat\\opl\\project1\\data\_for\_python.xlsx', index=False)

d = df.groupby([k]).size().reset\_index(name='Count')

for l in range(len(d)): # counter for each state in an observable char

with open("C:\\Users\\rehat\\opl\\project1\\part.dat",'w') as f:

f.write('NR\_Goods=2;\n'+'SheetConnection comm("data\_for\_python.xlsx");\n')

f.write('NR\_Observations= ' + str(d['Count'][l]) + ';\n')

if l==0:

f.write("""AllP from SheetRead(comm,"'Sheet1'!K2:L"""+ str(d['Count'][l]+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!I2:J"""+ str(d['Count'][l]+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!M2:M"""+ str(d['Count'][l]+1)+"""");\n""")

n=d['Count'][l]

elif l>0:

f.write("""AllP from SheetRead(comm,"'Sheet1'!K"""+str(n+2)+""":L"""+ str(n+d['Count'][l]+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!I"""+str(n+2)+""":J"""+ str(n+d['Count'][l]+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!M"""+str(n+2)+""":M"""+ str(n+d['Count'][l]+1)+"""");\n""")

n=n+d['Count'][l]

f.write('SheetConnection comm2("time\_allocation.xlsx");\n')

f.write("""NR\_Types to SheetWrite(comm2,"'Sheet'!"""+str(chr(c))+""""""+str(i)+""":"""+str(chr(c))+""""""+str(i)+"""");\n""")

f.write("""NR\_Observations to SheetWrite(comm2,"'Sheet'!D"""+str(i)+""":D"""+str(i)+"""");\n""")

f.write('runtype = "'+str(k)+'";\n')

f.write("""runtype to SheetWrite(comm2,"'Sheet'!A"""+str(i)+""":A"""+str(i)+"""");\n""")

f.write('runtype2 = "'+str(d[k][l])+'";\n')

f.write("""runtype2 to SheetWrite(comm2,"'Sheet'!B"""+str(i)+""":B"""+str(i)+"""");\n""")

call(["oplrun.exe", "C:\\Users\\rehat\\opl\\project1\\part.mod", "C:\\Users\\rehat\\opl\\project1\\part.dat"])

i=i+1 # increment by 1 at the end of each run

n=0 # reset to 0 at the end of k

l=0 # reset to 0 at the end of k

'''

3bis - loop over k and l given j (find tau\_k,j)

'''

for j in df[['Gender','Age','Education','Marital Status','Employment']]:

for k in df[['Gender','Age','Education','Marital Status','Employment']]:

# there should be an elegant way to do this but let's do it manually since it's just 5 choose 2

if j == 'Age' and k == 'Gender':

continue

if j == 'Education' and (k == 'Gender' or k == 'Age'):

continue

if j == 'Marital Status' and (k == 'Gender' or k == 'Age' or k == 'Education'):

continue

if j == 'Employment' and (k == 'Gender' or k == 'Age' or k == 'Education' or k == 'Marital Status'):

continue

if j == k:

continue

df = df.sort\_values([j,k])

df.to\_excel('C:\\Users\\rehat\\opl\\project1\\data\_for\_python.xlsx', index=False)

d = df.groupby([j,k]).size().reset\_index(name='Count')

for l in range(len(d)): # counter for each state in an observable char

with open("C:\\Users\\rehat\\opl\\project1\\part.dat",'w') as f:

f.write('NR\_Goods=2;\n'+'SheetConnection comm("data\_for\_python.xlsx");\n')

f.write('NR\_Observations= ' + str(d['Count'][l]) + ';\n')

if l==0:

f.write("""AllP from SheetRead(comm,"'Sheet1'!K2:L"""+ str(d['Count'][l]+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!I2:J"""+ str(d['Count'][l]+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!M2:M"""+ str(d['Count'][l]+1)+"""");\n""")

n=d['Count'][l]

elif l>0:

f.write("""AllP from SheetRead(comm,"'Sheet1'!K"""+str(n+2)+""":L"""+ str(n+d['Count'][l]+1)+"""");\n""")

f.write("""AllQ from SheetRead(comm,"'Sheet1'!I"""+str(n+2)+""":J"""+ str(n+d['Count'][l]+1)+"""");\n""")

f.write("""Income from SheetRead(comm,"'Sheet1'!M"""+str(n+2)+""":M"""+ str(n+d['Count'][l]+1)+"""");\n""")

n=n+d['Count'][l]

f.write('SheetConnection comm2("time\_allocation.xlsx");\n')

f.write("""NR\_Types to SheetWrite(comm2,"'Sheet'!"""+str(chr(c))+""""""+str(i)+""":"""+str(chr(c))+""""""+str(i)+"""");\n""")

f.write("""NR\_Observations to SheetWrite(comm2,"'Sheet'!D"""+str(i)+""":D"""+str(i)+"""");\n""")

f.write('runtype2 = "'+str(d[j][l])+' and ' +str(d[k][l])+ '";\n')

f.write("""runtype2 to SheetWrite(comm2,"'Sheet'!B"""+str(i)+""":B"""+str(i)+"""");\n""")

f.write('runtype = "'+str(j)+' and ' +str(k)+'";\n')

f.write("""runtype to SheetWrite(comm2,"'Sheet'!A"""+str(i)+""":A"""+str(i)+"""");\n""")

call(["oplrun.exe", "C:\\Users\\rehat\\opl\\project1\\part.mod", "C:\\Users\\rehat\\opl\\project1\\part.dat"])

i=i+1 # increment by 1 at the end of each run

n=0 # reset to 0 at the end of k

l=0 # reset to 0 at the end of k

df3bis.append(pd.read\_excel('C:\\Users\\rehat\\opl\\project1\\time\_allocation.xlsx', header=None))

'''

4th step - calculate kappa ratios and report the results for money allocation task

'''

resultcopy = df3[:] # copy of results so things do not get lost

df5 = pd.DataFrame() # intermediate dataframe object combining best 2-level values

x=0

for x in range(len(df3)):

df3[x] = df3[x].rename(columns={3:'n'})

df3[x] = df3[x].rename(columns={2:'Types'})

df3[x] = df3[x].rename(columns={1:'State'})

df3[x] = df3[x].rename(columns={0:'Obs\_Char'})

x=0

for x in range(len(df3)):

df3[x]['Sum\_Types'] = df3[x].groupby(['Obs\_Char'])['Types'].transform('sum')

df3[x]['Ratio\_'+ str(x+1)] = df3[x]['Sum\_Types']/df3[x]['Sum\_Types'][0]

df3[x]= df3[x].sort\_values('Obs\_Char')

df5 = df5.append(df3[x])

with pd.ExcelWriter('C:\\Users\\rehat\\opl\\project1\\money\_allocation.xlsx', engine="openpyxl", mode='a', if\_sheet\_exists='new') as writer:

df3[x].to\_excel(writer, header=["Obs\_Char", "State", "Types", "n", "Sum\_Types", "Ratio"], index=False, sheet\_name= str(x+1))

df5 = df5.drop(['Types', 'n', 'Sum\_Types'], 1)

df5 = df5.reset\_index(drop=True)

df5 = df5.groupby(['Obs\_Char', 'State'], as\_index=False).sum()

df5 = df5.replace(0, np.nan)

''' Way to find the best ratio (not accounting for equalities btwn ratios) - not useful anymore

# for x in range(len(df3)):

# df3[x]['Sum\_Types'] = df3[x].groupby(['Obs\_Char'])['Types'].transform('sum')

# df3[x]['Ratio'] = df3[x]['Sum\_Types']/df3[x]['Sum\_Types'][0]

# df3[x]= df3[x].sort\_values('Ratio')

# df4[x] = df3[x][df3[x].Obs\_Char.str.contains('and')][:1]

# df5 = df5.append(pd.DataFrame(list(zip(df4[x]['Obs\_Char'], df4[x]['Ratio'])), columns =['Obs\_Char', 'Ratio']))

# df5 = df5.reset\_index(drop=True)

# with pd.ExcelWriter('C:\\Users\\rehat\\opl\\project1\\money\_allocation.xlsx', engine="openpyxl", mode='a', if\_sheet\_exists='new') as writer:

# df3[x].to\_excel(writer, header=["Obs\_Char", "State", "Types", "n", "Sum\_Types", "Ratio"], index=False, sheet\_name= str(x+1))

'''

with pd.ExcelWriter('C:\\Users\\rehat\\opl\\project1\\money\_allocation.xlsx', engine="openpyxl", mode='a', if\_sheet\_exists='new') as writer:

df5.to\_excel(writer, header=df5.columns, index=False, sheet\_name= 'Results Matrix')

'''

4bis - calculate kappa ratios and report the results for time allocation task

'''

resultcopybis = df3bis[:] # copy of results so things do not get lost

df5bis = pd.DataFrame() # intermediate dataframe object combining best 2-level values

x=0

for x in range(len(df3bis)):

df3bis[x] = df3bis[x].rename(columns={3:'n'})

df3bis[x] = df3bis[x].rename(columns={2:'Types'})

df3bis[x] = df3bis[x].rename(columns={1:'State'})

df3bis[x] = df3bis[x].rename(columns={0:'Obs\_Char'})

x=0

for x in range(len(df3bis)):

df3bis[x]['Sum\_Types'] = df3bis[x].groupby(['Obs\_Char'])['Types'].transform('sum')

df3bis[x]['Ratio\_'+ str(x+1)] = df3bis[x]['Sum\_Types']/df3bis[x]['Sum\_Types'][0]

df3bis[x]= df3bis[x].sort\_values('Obs\_Char')

df5bis = df5bis.append(df3[x])

with pd.ExcelWriter('C:\\Users\\rehat\\opl\\project1\\time\_allocation.xlsx', engine="openpyxl", mode='a', if\_sheet\_exists='new') as writer:

df3bis[x].to\_excel(writer, header=["Obs\_Char", "State", "Types", "n", "Sum\_Types", "Ratio"], index=False, sheet\_name= str(x+1))

df5bis = df5bis.drop(['Types', 'n', 'Sum\_Types'], 1)

df5bis = df5bis.reset\_index(drop=True)

df5bis = df5bis.groupby(['Obs\_Char', 'State'], as\_index=False).sum()

df5bis = df5bis.replace(0, np.nan)

with pd.ExcelWriter('C:\\Users\\rehat\\opl\\project1\\time\_allocation.xlsx', engine="openpyxl", mode='a', if\_sheet\_exists='new') as writer:

df5bis.to\_excel(writer, header=df5.columns, index=False, sheet\_name= 'Results Matrix')

'''

CONCLUDING INFORMATION

'''

elapsed = time.time() - t

print('Computation done with ' + str(nb\_draws) + ' subsamples whose size equals ' + str(sample\_size\*100) + ' percent of the original data. \nTotal elapsed time (in seconds): '

+ str(elapsed) + '\nAverage loop length (in seconds): ' + str(elapsed/nb\_draws))