Finance of Emerging Markets, Tutorial Week 3: Legal origin and financial development

In question 1, you will build a simple Python program to calculate the interest rate of a loan. You could use the loan pricing model we discussed in the lecture.

Question 1.a: What is the difference between a string and a floating variable?

String variables are variables that contain not just numbers, but also other characters.

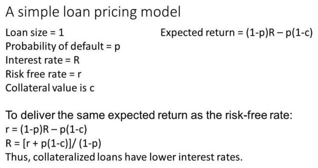
A floating variables are variables that can hold a real number, such as 4566, -3.66, or 0.0777.

You cannot make calculations with string variables, even if the string only includes numbers. However, it is possible to convert a numerical string variable into a floating variable using the “float()” command.

Question 1.b: Create a code requesting the user to input the probability of default, risk-free rate, collateral value (= collateral value / loan size). Use the commands “input()”, “float()”, and “print()”. You could use the “Covered Interest Parity” program (on QMPlus) as inspiration.

```
#Inputs
prob_default = input('Enter probability of default (e.g. 0.06) \n')
prob_default = float(prob_default)
risk_free_rate = input('Enter risk-free rate (e.g. 0.02) \n')
risk_free_rate = float(risk_free_rate)
collateral_value = input('Enter collateral value (e.g. 0.80) \n')
```

Question 1.c: Calculate the interest rate using the loan pricing model. Print the outcome.



A simple loan pricing model

Loan size = 1

Probability of default = p

Interest rate = R

Risk free rate = r

Collateral value is c

Expected return = (1-p)R – p(1-c)

To deliver the same expected return as the risk-free rate:

r = (1-p)R – p(1-c)

R = [r + p(1-c)]/ (1-p)

Thus, collateralized loans have lower interest rates.

```
#Calculate the interest rate

interest_rate = (risk_free_rate + prob_default * (1 - collateral_value))/(1 - prob_default)

print("Interest rate: " + "%.3f M " % interest_rate )
```

Question 1.d: (Example exam question lecture) Country A has an expected recovery rate on collateral of 0.8 and country B an expected recovery rate of 0.6. The loan amount is 1, the risk-free rate 0.02, and the probability of default of firms in both countries is 0.06. What is the interest differential between these two countries if loans yield the same expected return as the risk-free rate.

Country A: Interest rate 0.034

Country B: Interest rate 0.047

Interest rate differential = 0.013

Question 1.e: Plot the interest rate as a function of the collateral value. Use loan pricing model formula and the following piece of code. Replace the parts in red:

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(0,1)
y = 2 * x + 5
plt.xlabel("x axis caption")
plt.ylabel("y axis caption")
plt.plot(x,y)
plt.show()
```

```
import matplotlib.pyplot as plt

import numpy as np

# 10 linearly spaced numbers

x = np.linspace(0,1)

# the function, which is y = x^2 here

y = (risk_free_rate + prob_default * (1 - x))/(1 - prob_default)

# setting the axes at the centre

fig = plt.figure()
```

Question 2: The objective of this question is to learn how to visualize data using Python. We will replicate the map of last lecture including the legal origin of all countries.

To create the map, you need to download the data of La Porta et al. (JEL2008), The Economic Consequences of Legal Origin, and this "shape file". Save them in the same folder.

You also need to install the package geopandas. You will find a video on QMPlus explaining how to install this package.

Question 2.a: Load the Legal Origin data using the following code. Replace the parts in red:

```
import json

import pandas as pd

import geopandas as gpd

import matplotlib.pyplot as plt


# Set the folder of the files (Use: /, not \)

file_path = "C:/Users/thoma/Dropbox/Queen Mary/"

excel_file = file_path + "economiccon_data.xls"

#Load the excel data into python

df = pd.read_excel (open(excel_file,'rb'), sheet_name='Table 1')

print (df)
```

Question 2.b: Load and plot the shape file using the code below.

```
# Setting the path to the shapefile

SHAPEFILE = file_path + 'ne_10m_admin_0_countries.shp'

# Read shapefile using Geopandas

geo_df = gpd.read_file(SHAPEFILE)[['ADMIN', 'ADM0_A3', 'geometry']]

# Rename columns.

geo_df.columns = ['country', 'country_code', 'geometry']

geo_df.head(3)

# Drop row for 'Antarctica'. It takes a lot of space in the map and
is not of much use

geo_df = geo_df.drop(geo_df.loc[geo_df['country'] ==
'Antarctica'].index)

# Print the map

geo_df.plot(figsize=(20, 20), edgecolor='white', linewidth=1,
color='lightblue')
```

Question 2.c: Merge the legal origin data with the shape file using the code below. In the tutorial I will explain the merging function.

```
#Data cleaning. Make sure that the matching variables have identical
identifiers
for index, item in enumerate(geo_df['country_code']):
    if item=="ROU":
        geo_df['country_code'][index] = "ROM"
    if item=="COD":
        geo_df['country_code'][index] = "ZAR"
    if item=="SRB":
        geo_df['country_code'][index] = "YUG"
    if item=="MNE":
        geo_df['country_code'][index] = "YUG"

# Merge the two dataframes
merged_df = pd.merge(left=geo_df, right=df, how='left',
left_on='country_code', right_on='code')
```

Question 2.d: Create a new veriable “legal origin” using the code below.

```
#Create a legal origin variable
merged_df = merged_df.assign(legal_origin = 0)
for index, item in enumerate(merged_df['legor_uk']):
    if item==1:
        merged_df['legal_origin'][index] = 1
for index, item in enumerate(merged_df['legor_fr']):
    if item==1:
        merged_df['legal_origin'][index] = 2
for index, item in enumerate(merged_df['legor_ge']):
    if item==1:
        merged_df['legal_origin'][index] = 3
for index, item in enumerate(merged_df['legor_sc']):
    if item==1:
        merged_df['legal_origin'][index] = 4
for index, item in enumerate(merged_df['legor_so']):
    if item==1:
        merged_df['legal_origin'][index] = 5
```

Question 2.e: Plot the legal origin map using the code below.

```
title = 'Legal origins'
clusdict={1: 'English', 2: 'French', 3: 'German', 4:
'Scandinavian',5: 'Socialist', 0: 'No data'}
fig = plt.figure(figsize=(15, 15))
ax = plt.gca()
ax.set_title(title, fontdict={'fontsize': '25', 'fontweight': '3'})
merged_df.plot(ax=ax, edgecolor='white',cmap='viridis', linewidth=1,
column='legal_origin', legend=True, categorical=True,
legend_kwds={'loc': 'lower left'});
def replace_legend_items(legend, mapping):
    for txt in legend.get_texts():
        for k,v in mapping.items():
            if txt.get_text() == str(k):
                txt.set_text(v)
replace_legend_items(ax.get_legend(), clusdict)
plt.show()
```