# 文件名  
shfeDataFileName = 'shfeAu20160101\_20190111.csv'  
shfeIndexDataFileName = 'SHFEAuIndex.csv'  
xauUsdDataFileName = 'xauusd20160101\_20190111.csv'  
forexCNYUSDFileName = 'CNYUSDforex.csv'

# 然后开始建模  
class getVarModel():  
 # def \_\_init\_\_(self, shfeAuIndexFileName=shfeIndexDataFileName, xauusdFileName=xauUsdDataFileName, inSampleStart='2016-1-4',inSampleEnd='2018-9-28'):  
  
 def \_\_init\_\_(self, shfeAuIndexFileName=shfeIndexDataFileName, xauusdFileName=xauUsdDataFileName,  
 inSampleStart='2018-1-1',inSampleEnd='2018-9-28'):  
 self.shfeIndexData = pd.read\_csv(shfeAuIndexFileName)  
 self.xauusdData = pd.read\_csv(xauusdFileName)  
 self.forexData = pd.read\_csv(forexCNYUSDFileName)  
 # 处理数据，用日期做index  
 self.shfeIndexData[dateTimeName] = pd.to\_datetime(self.shfeIndexData[dateTimeName])  
 self.shfeIndexData = self.shfeIndexData.set\_index(dateTimeName)  
 self.xauusdData[dateTimeName] = pd.to\_datetime(self.xauusdData[dateTimeName])  
 self.xauusdData = self.xauusdData.set\_index(dateTimeName)  
 self.forexData[dateTimeName] = pd.to\_datetime(self.forexData[dateTimeName])  
 self.forexData = self.forexData.set\_index(dateTimeName)  
 # 样本内数据,构造Series类型数据  
 self.shfeIndexInSampleData = self.shfeIndexData[inSampleStart:inSampleEnd]  
 self.shfeIndexInSampleData = pd.Series(self.shfeIndexInSampleData[indexPriceName],  
 index=self.shfeIndexInSampleData.index)  
 self.xauusdDataInSampleData = self.xauusdData[inSampleStart:inSampleEnd]  
 self.xauusdDataInSampleData = pd.Series(self.xauusdDataInSampleData[xauusdPriceName],  
 index=self.xauusdDataInSampleData.index)  
 self.forexDataInSampleData = self.forexData[inSampleStart:inSampleEnd]  
 self.forexDataInSampleData = pd.Series(self.forexDataInSampleData['OPEN'],  
 index=self.forexDataInSampleData.index)  
 # 样本个数  
 self.sampleNum = self.shfeIndexInSampleData.shape[0]  
  
 # 价格走势图  
 def price\_trend\_draw(self):  
 # 处理数据,化为人民币和1g  
 # changeXAUUSD = np.array(self.xauusdDataInSampleData) / 28.3495 \* 6.78  
 font1 = {'size':23}  
 changeXAUUSD = np.multiply(np.array(self.xauusdDataInSampleData) / 28.3495 , np.array(self.forexDataInSampleData))  
 changeXAUUSD = np.round(changeXAUUSD,2)  
 shfeXAU = np.array(self.shfeIndexInSampleData)  
 # 算相关系数，一定要是pd.Series类型数据才可以  
 correlation = round(pd.Series(shfeXAU).corr(pd.Series(changeXAUUSD)),6)  
 # print(changeXAUUSD)  
 fig = plt.figure(figsize=(12,8))  
 plt.plot(changeXAUUSD,'r',label='XAU USD')  
 plt.plot(shfeXAU,'g',label='SHFE XAU')  
 plt.title('Correlation: ' + str(correlation),font1)  
 plt.grid(True)  
 plt.axis('tight')  
 plt.legend(loc=0,prop=font1)  
 plt.ylabel('Price',font1)  
 plt.show()  
 # plt.plot()  
  
 # 对数一阶差分处理  
 def logdiff(self,data):  
 logData = np.log(data)  
 logDiffData = np.diff(logData)  
 return logDiffData  
 # 数据稳定性检验  
 def adftest(self,data,maxlags):  
 adfResult = sm.tsa.stattools.adfuller(data,maxlags)  
 output = pd.DataFrame(index=['Test Statistic Value', "p-value", "Lags Used", "Number of Observations Used",  
 "Critical Value(1%)", "Critical Value(5%)", "Critical Value(10%)"],  
 columns=['value'])  
 output['value']['Test Statistic Value'] = adfResult[0]  
 output['value']['p-value'] = adfResult[1]  
 output['value']['Lags Used'] = adfResult[2]  
 output['value']['Number of Observations Used'] = adfResult[3]  
 output['value']['Critical Value(1%)'] = adfResult[4]['1%']  
 output['value']['Critical Value(5%)'] = adfResult[4]['5%']  
 output['value']['Critical Value(10%)'] = adfResult[4]['10%']  
 return output  
 # 建立var模型,初次先多次使用，查看aic，bic  
 def buildModle(self,dataframe,varLagNum):  
 orgMod = sm.tsa.VARMAX(dataframe,order=(varLagNum,0),trend='c',exog=None)  
 fitMod = orgMod.fit(maxiter=1000,disp=False)  
 print(fitMod.summary())  
 resid = fitMod.resid  
  
 result = {'fitMod':fitMod,'resid':resid}  
 # 最后返回mod用来做后面一系列检验  
 return result  
 # VAR系统平稳性检验（代替eviews单位根AR root图）  
 def olsCusum(self,resid):  
 # 原假设：无漂移（平稳），备择假设：有漂移（不平稳）  
 result = statsmodels.stats.diagnostic.breaks\_cusumolsresid(resid)  
 print(result)  
 return result  
 # 协整  
 def cointTest(self,data1,data2):  
 result = sm.tsa.stattools.coint(data1,data2)  
 print(result)  
 # 脉冲响应  
 def impulseResponse(self,fitMod,terms=20):  
 font1 = {'size':23}  
 ax = fitMod.impulse\_responses(terms, orthogonalized=True).plot(figsize=(12, 8))  
 plt.legend(prop=font1)  
 plt.show()  
 def varianceDue(self,dataFrame):  
 font1 = {'size': 23}  
 md = sm.tsa.VAR(dataFrame)  
 re = md.fit(2)  
 fevd = re.fevd(10)  
 print(fevd.summary())  
 fevd.plot()  
 plt.legend(prop=font1)  
 plt.show()  
  
 def runModle(self):  
 # 画价格走势图  
 # self.price\_trend\_draw()  
 # 一阶对数差分,把数据的日期索引留出来  
 lnSHFEDiff = self.logdiff(self.shfeIndexInSampleData)  
 lnSHFEDiffIndex = self.shfeIndexInSampleData.index[1:]  
 lnXAUDiff = self.logdiff(self.xauusdDataInSampleData)  
 lnXAUDiffIndex = self.xauusdDataInSampleData[1:]  
 # print(len(lnSHFEDiffIndex))  
 # print(lnSHFEDiff)  
 # adf 一次检验  
 # shfeAdfResult = self.adftest(self.shfeIndexInSampleData,4)  
 # print(shfeAdfResult)  
 # xauAdfResult = self.adftest(self.xauusdDataInSampleData,4)  
 # print(xauAdfResult)  
 # adf 二次检验  
 shfeAdfResult = self.adftest(lnSHFEDiff,10)  
 # print(shfeAdfResult)  
 xauAdfResult = self.adftest(lnXAUDiff,10)  
 # print(xauAdfResult)  
 # 建模，1-5的之后项  
 # 只能用dataFrame多维格式  
 lnDataDict = {'lnSHFEDiff':lnSHFEDiff,'lnXAUDiff':lnXAUDiff}  
 lnDataDictSeries = pd.DataFrame(lnDataDict,index=lnSHFEDiffIndex)  
 data = lnDataDictSeries[['lnSHFEDiff','lnXAUDiff']]  
 # 协整检验  
 self.cointTest(lnDataDictSeries['lnSHFEDiff'], lnDataDictSeries['lnXAUDiff'])  
 # 建模  
 shfexau\_xauusdMode = self.buildModle(data,2)  
 fitMod = shfexau\_xauusdMode['fitMod']  
 shfeResid = shfexau\_xauusdMode['resid']['lnSHFEDiff']  
 xauusdResid = shfexau\_xauusdMode['resid']['lnXAUDiff']  
 # cusum检验  
 # self.olsCusum(shfeResid)  
 # 脉冲响应(默认使用乔里斯基正交)  
 self.impulseResponse(fitMod)  
 # ax = fitMod.impulse\_responses(10, orthogonalized=True)  
 # 方差风险  
 self.varianceDue(data)  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # 生成指数文件  
 # 建模  
 varmodel = getVarModel()  
 varmodel.runModle()  
  
 # a = '2019/1/11'  
 # aMonth = arrow.get(a).month  
 # print(aMonth,type(aMonth))