Vrije Universiteit Amsterdam

Econometrics for Quantitative Risk Management

Forecasting log-volumes of Apple Stock by Prediction Error Decomposition and Maximum Likelihood Estimation

Block 2 Assignment I

Authors
Aytek Mutlu (2648232) & Cindy Zegers (2561482)

November 10, 2019

1 Introduction

Daily volume is an important component of market structure, especially in the areas of portfolio management and high frequency trading. For example, asset prices might be affected when large amounts of orders are executed during a short period of time, which for portfolio managers translates into a risk, as they constantly need to acquire or liquidate positions. This risk entails a cost that could their lower profits. Predicting daily volume could mitigate this loss by reducing the costs entailed with the higher risk. In addition to portfolio managers, high frequency traders also benefit from predicting daily volumes. Namely, it allows them to take advantage of arbitrage opportunities before investors' expectations are truly reflected in prices. This will result in higher profits. As predicting volumes could have advantages in multiple areas, it is an important element in financial markets. For that reason, we have estimated 4 different models predicting Apple stock's daily log volumes.

2 Methodology

2.1 Data

The dataset contains a total of 7559 observations of daily volumes of Apple stock in a period from 3rd March 1989 till 31st December 2018, which we converted to log volumes. As there are no missing values, we could use all observations. The daily log volumes of Apple stock are graphically presented in Figure 1. There is one spike around 2001, but since this is still smaller than the maximum value of daily log volumes, there seem to be no outliers. The corresponding summary statistics of daily log volumes of Apple stock are presented in Table 1. As shown in the table, the mean daily log volume is 15.844 and the standard deviation of daily log volumes is 1.318. The distribution of daily log volumes is not skewed or fat-tailed as can be seen from a value for skewness of -0.122 and a value for kurtosis of -1.184. Taking into account the values of all summary statistics, there is no reason to believe if there are any outliers; however, to have a reliable method to determine if there are outliers, we have done additional testing. We have tested if there are outliers by performing the Tukey's test for outliers with k=1.5. Based on this test, there are no outliers in our dataset, hence, we have not done any cleaning to remove any outliers. This means that the original dataset containing 7559 observations of daily volume is used to estimate a model that predicts Apple's daily log volumes.

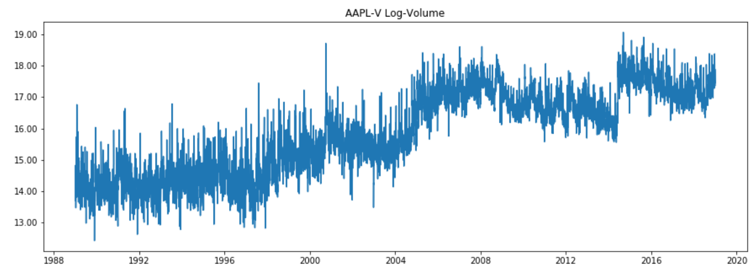

Figure 1: **Apple stock daily log volumes from 3rd March 1989 till 31st December 2018.**The figure shows daily log volumes of Apple stock from the period 3rd March 1989 till 31st December 2018. The values on the y-axis represent daily log volumes, and those values are obtained by transforming the original observed daily volumes of Apple stock into daily log volumes. The horizontal axis corresponds to the time period over which the data is observed.

Table 1

Summary statistics of Apple stock daily log volumes

This table reports the mean, standard deviation, minimum, maximum, skewness, and kurtosis of Apple stock daily log volumes based on data in the period from 3rd March 1989 till 31st December 2018. The summary statistics are based on a total of 7559 daily observations of Apple stock log volumes. There are no missing values in the dataset.

Descriptive	**Value**
Mean	15.844
Standard deviation	1.318
Minimum	12.426
Maximum	19.062
Skewness	-0.122
Kurtosis	-1.184

2.2 Estimation

The total time period of available data is split up into 3 different periods; namely, the 1990's, the 2000's, and the 2010's. For each of these time periods, 4 different ARMA(p,q) models with intercept and linear trend have been estimated, by which p and q take value 0 or 1. This means that for every time period we have estimated an ARMA(0,0) model, an ARMA(0,1) model, an ARMA(1,0) model, and an ARMA(1,1) model. Written in equation format, the have estimated the following model:

$$y_t = \mu + \phi_1 * y_{t-1} + \epsilon_t + \theta_1 * \epsilon_{t-1} \tag{1}$$

For values of p and q equal to 0 or 1, we have estimated the above model by the use of Prediction Error Decomposition and Maximum Likelihood estimation with the assumption of Normally distributed disturbance terms with mean 0 and variance σ^2, i.e., ϵ_t $N(0, \sigma^2)$. In addition, we have assumed weakly stationarity of the dependant variable (daily log volumes).

For each estimated model in all 3 time periods, we have obtained 4 different estimates; namely, an estimate for intercept (μ in the equation), an estimate for ϕ_1, an estimate for θ_1, and an estimate for the variance (which is σ^2). Based on the optimal parameter values, we have calculated the AIC and BIC for each model in all time periods. Within a time period, the ARMA(p,q) model with the lowest AIC has been selected as the best model for that time period. Furthermore, we have also used Statsmodels to obtain the optimal parameter values and AIC/BIC values for each model in all 3 time periods. We then assessed whether there are differences between the estimates and AIC/BIC values obtained by ML estimation versus the estimates and AIC/BIC values obtained by using Statsmodels.

To avoid ending up with a estimates for a local maximum instead of global maximum, we first estimated the ARMA(0,0) model. Then we used the converged parameter values of this model as starting values for the ARMA(0,1) model and the ARMA(1,0) model. Then we used the converged parameter values of the ARMA(0,1) model as starting values for the ARMA(1,1) model.

3 Results

Table 2

AIC and BIC values for 4 ARMA(p,q) models in 3 different decades

This table reports the AIC and BIC values of an ARMA(0,0) model, an ARMA(0,1) model, an ARMA(1,0) model, and an ARMA(1,1) model, respectively, all with intercept and linear trend. The AIC and BIC values for these 4 different ARMA models are calculated for 3 different time periods; namely, the 1990's, the 2000's, and the 2010's. The dependent variable y_t represents the daily log volume of Apple stock. By the use of Prediction Error Decomposition and Maximum Likelihood optimization, parameter estimates are obtained for all parameters in Equation 1. Based on these optimal estimates, the AIC and BIC has been calculated for each ARMA model in each time period. Within a time period, the bold AIC value corresponds to the lowest AIC value for that time period.

	The 1990's		**The 2000's**		**The 2010's**	
	AIC	**BIC**	**AIC**	**BIC**	**AIC**	**BIC**
ARMA(0,0)	6048.779	6060.64	7334.443	7346.293	4036.441	4047.891
ARMA(0,1)	4695.728	4713.519	5182.307	5200.083	2577.92	2595.095
ARMA(1,0)	3822.272	3840.062	2518.521	2536.297	1192.7	1209.875
ARMA(1,1)	**3738.764**	3762.485	**2203.938**	2227.64	**1044.091**	1066.99

As presented in Table 2, the lowest AIC value is found for the ARMA(1,1), regardless of the estimated time period. In the table these values are shown in bold. The ARMA(1,1) model seems to estimate the 2010's best of 3 decades, as this is results in the lowest AIC value. For all 3 decades, the ARMA(0,0) gives the highest AIC value, followed by the ARMA(0,1) model, then followed by the ARMA(1,0) model.

For the ARMA(1,1) model, the estimates and robust standard errors of the parameter values based on PED and ML estimation on one hand, and by using Statsmodels on the other hand, are shown in Table 3 below. All parameter estimates obtained by PED and ML estimation are close if not equal to the estimates obtained by using Statsmodels. The largest differences between the estimates of the two methods is found in the estimate for μ. These differences could have arisen because of a different optimization function; specifically, the optimization function used for PED and ML estimation does not include the log density function for the first observation, whereas the optimization function used by the package does include this log density function. Also the standard errors of μ differ between the computation method versus the package; in particular, the standard errors of μ based on the computation method are about 6 times larger compared to the standard error of μ based on the package. This is due to the fact that package reports the unconditional mean and the standard error of unconditional mean. For comparison purposes, mean reported by package is converted to the intercept and reported as such. Yet, standard error is not converted and therefore reported as standard error of the unconditional mean. The other 3 parameter estimates have much lower standard errors, both based on computation and package, indicating these estimates are more precise. Regarding those estimates, the estimates for ϕ_1 is positive for each decade with a value of 0.875, 0.981, and 0.945 for the 1990's, the 2000's, and the 2010's, respectively. This means that, other things equal, there is substantial memory in this model, and because of the fact that ϕ_1 is close to, but smaller than 1, spikes will go down slowly over time instead of instantly. On the other hand, the estimates for θ_1 are all negative with a value of -0.320, -0.529, and -0.403 for the 1990's, the 2000's, and the 2010's, respectively. This indicates that, other things equal, the disturbance on day t-1 directly lowers the daily log volume on day t by around a third, around a half, and around 40%, respectively. Regarding σ^2, we obtained estimates of 0.224, 0.129, and 0.093 for the 1990's, the 2000's, and the 2010's, respectively. This indicates that the smallest variation corresponds to the 2010's given that we estimated an ARMA(1,1) model. The same result is found when taking the AIC value into account; this value is also lowest for the 1990's regardless of the estimated model.

Table 3
Parameter estimates and robust standard errors for the ARMA(p,q) model with the lowest AIC in each of the 3 different time periods
This table report the estimated parameter values for the ARMA(1,1) model for 3 different periods; namely, the 1990's, the 2000's, the 2010's. For each time period, the left column of estimates are found by Prediction Error Decomposition and Maximum Likelihood estimation. The right column of estimates for a given time period are obtained using statsmodels. For values of p and q both equal to 1, statsmodels resulted in the lowest AIC value, and it's this model the estimates are based on. The parameter estimates are obtained for all parameters in Equation 1. The standard errors of the estimates are shown in brackets and those are robust standard errors obtained by computing the product of the inverse Hessian matrix, the Jacobian, and the inverse Hessian matrix, respectively. Standard errors reported for σ^2 are actually standard errors of σ as that parameter is estimated in the model. Yet, the value of σ^2 is reported for comparison purposes. It should be noted that statsmodels package does not provide standard errors for σ^2.

	The 1990's		**The 2000's**		**The 2010's**	
	Computation	**Package**	**Computation**	**Package**	**Computation**	**Package**
μ	1.828 (0.263)	1.833 (0.048)	0.315 (0.085)	0.314 (0.165)	0.93 (0.192)	0.936 (0.069)
ϕ_1	0.875 (0.018)	0.874 (0.014)	0.981 (0.005)	0.981 (0.004)	0.945 (0.011)	0.945 (0.009)
θ_1	-0.32 (0.050)	-0.32 (0.034)	-0.529 (0.047)	-0.529 (0.027)	-0.403 (0.054)	-0.403 (0.032)
σ^2	0.224 (0.009)	0.224	0.129 (0.007)	0.13	0.093 (0.006)	0.093
AIC	3738.764	3739.79	2203.938	2207.133	1044.091	1045.704

4 Conclusion

In conclusion, based on daily log volumes of Apple stock from the period of 3rd March 1989 till 31st December 2018, 4 different ARMA(p,q) models have been estimated for 3 decades; namely, the 1990's, the 2000's, and the 2010's. Estimation is based on Prediction Error Decomposition and Maximum Likelihood estimation with the assumption of Normally distributed disturbance terms and weakly stationarity of the dependent variable. For each decade, the model that resulted in the lowest AIC value was the ARMA(1,1) model. Moreover, the ARMA(1,1) model fits the 2010's best of 3 decades, as this decade corresponds to the lowest AIC value. In addition to the estimation method based on computation, Statsmodels is used as reference to estimate the same parameters as in the computation method. The parameter estimates based on both methods produced almost, if not completely, identical values. For each decade, a value between about 0.8 and 1 was found for ϕ_1, and a value between about -0.55 and -0.3 was found for θ_1, and a value between about 0.09 and 0.25 was found for σ^2. Regarding the latter, the lowest estimate of σ^2 corresponds to the 2010's, which is in correspondence with the lowest AIC value. Taking those into account, we concluded that there is substantial memory in the model when predicting Apple stock's daily log volumes, that the disturbance terms of the previous day have a direct decreasing effect on daily log volumes today, and that the estimated ARMA(1,1) model predicts the daily log volumes of Apple stock best for the 2010's.

Appendices

A Source Code

```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 28 09:47:54 2019

@author: aytekmutlu & cindyzegers
"""

import pandas as pd
import numpy as np
from scipy.stats import describe
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter
import scipy.optimize as opt
from statsmodels.tsa.arima_model import ARMA
import statsmodels.api as sm
import math
import copy

from lib.grad import *

def read_clean_data(stocks,filename):
    '''
    Purpose:
        Read and clean volume data of given stocks

    Inputs:
        stocks:     list, with list of stocks
        filename:   string, filename for volumes

    Return value:
        df          dataframe, data
    '''
    ## read data and extract volume columns
    df  = pd.read_csv('data/'+filename+'.csv', index_col='date',
    parse_dates=['date'],dayfirst=False)
    df = df[[s+'-V' for s in stocks]]

    ##take natural log
    df = np.log(df)

    ##plot data initially
    df.plot(figsize=(10,3))

    ## DATA CLEANING

    #no missing values

    print(df.info())

    #histogram of data
    df.plot.hist(bins=30)
```

```

    #check outliers
    inter_quartile_range = (df.describe().loc['75%'] - df.describe().loc['25%']).values
    outlier_min_limit = df.describe().loc['25%'].values - 1.5*inter_quartile_range
    outlier_max_limit = df.describe().loc['75%'].values + 1.5*inter_quartile_range
    outliers = df[np.logical_or((df>outlier_max_limit).values,(df<outlier_min_limit).values)]
    #outliers is empty so there are no outliers according to Tukey test

    return df

def plot_summarize_data(df):
    '''
    Purpose:
        Plot and summarize data

    Inputs:
        df:                 dataframe, data

    Return value:
        df_summary          dataframe, descriptives
    '''
    ##plot volumes
    fig,ax = plt.subplots(figsize=(15,5))
    ax.plot(df)
    ax.yaxis.set_major_formatter(FormatStrFormatter('%.2f'))
    ax.set_title(df.columns[0]+' Log-Volume')
    plt.savefig('plots.png')


    ##summarize data
    df_desc = describe(df)
    df_summary = pd.DataFrame(columns = df.columns,index=['Mean','Std. Deviation','Min','Max','Skewness','Kurtosis'])

    df_summary.loc['Mean'] = df_desc[2]
    df_summary.loc['Std. Deviation'] = np.sqrt(df_desc[3])
    df_summary.loc['Min'] = df_desc[1][0]
    df_summary.loc['Max'] = df_desc[1][1]
    df_summary.loc['Skewness'] = df_desc[4]
    df_summary.loc['Kurtosis'] = df_desc[5]

    return df_summary

def LL_PredictionErrorDecomposition(vP,vY,p,q):
    '''
    Purpose:
        Calculating Log Likelihood with prediction error decomposition

    Inputs:
        vP:     list, parameters
        vY:     list, time-series
```

```
        p:      integer, AR parameter
        q:      integer, MA parameter

    Return value:
        LL:     float, log-likelihood
    '''

    #parameter decomposition
    m = vP[0]
    phis = vP[1:p+1]
    thetas = vP[p+1:p+q+1]
    sigma = vP[-1]

    p = len(phis)
    q = len(thetas)

    #number of observations
    iY = len(vY)

    #initiation of residuls matrix
    vE = np.zeros(iY)

    for i in range(iY):

        #contribution of AR terms to vE
        vEp = 0
        if i>=p:
            for j in range(1,p+1):
                vEp += vY[i-j]*phis[j-1]

        #contribution of MA terms to vE
        vEq = 0
        if i>=q:
            for j in range(1,q+1):
                vEq += vE[i-j]*thetas[j-1]

        #combine both effects to find vE
        if i>0:
            vE[i] = vY[i] - m  - vEp - vEq


    vLL = -0.5*(np.log(2*math.pi*(sigma**2)) + vE*vE/(sigma**2))

    return vLL


def ARMA_compute(df,stock,year,p,q,vP):
    '''
    Purpose:
        Building ARMA(p.q) model with log-likelihood with
    prediction error decomposition

    Inputs:
        df:            dataframe, data
        stock:         string, name of the stock to be analyzed
        year:          integer, start of the decade to be analyzed
        p:             integer, AR parameter
```

```
        q:              integer, MA parameter
        vP:             list, parameters

    Return values:
        scores:         list, AIC and BIC scores
        params:         list, m, phi, theta and sigma parameters
        std_errors:     list, std_errors of parameters
    '''
    #time-series to be modeled
    vY = df.loc[str(year):str(year+10),stock]

    #list of initial parameters based on p and q
    #in order to avoid local optima, some rules are set for initial parameters
    if (p==1) & (q==1) :
        vP0 = [vP[0],1,vP[2][0],vP[3]]
    elif (p==0) & (q==0):
        vP0 = [vP[0],vP[3]]
    elif (p==0) & (q==1):
        vP0 = [vP[0],1,vP[3]]
    elif (p==1) & (q==0):
        vP0 = [vP[0],1,vP[3]]


    #minimizing function
    sumLL= lambda vP: -np.sum(LL_PredictionErrorDecomposition(vP,vY,p,q))
    res= opt.minimize(sumLL, vP0, method="BFGS")
    print('Parameters are estimated by ML for '+stock+' for decade '+ str(year) + ' with p: '+str(p)+' and q: ' + str(q)+'\n')
    print('Optimization Success: ',res.success)

    #parameter estimates
    phis = res.x[1:p+1]
    m = res.x[0]
    thetas = res.x[p+1:p+q+1]
    sigma = res.x[-1]
    params = [m,phis,thetas,sigma]

    #scores
    aic = res.fun*2+2*len(res.x)
    bic = np.log(len(vY))*len(res.x) + 2*res.fun
    scores = [aic,bic]

    #hessian and std. errors
    hes = -hessian_2sided(sumLL,res.x)
    inv_hes = np.linalg.inv(hes)
    std_errors = list(np.sqrt(np.diag(-inv_hes)))

    #sandwich form robust std. errors
    inv_hes_symetric = (inv_hes + inv_hes.T)/2
    mG = jacobian_2sided(LL_PredictionErrorDecomposition,res.x,vY,p,q)

    cov_matrix_sandwich = inv_hes_symetric @ (mG.T @ mG) @ inv_hes_symetric
```

```
    std_errors_sandwich = list(np.sqrt(np.diag(cov_matrix_sandwich)
    ))


    return [scores,params,std_errors_sandwich],params


def ARMA_package(df,stock,year,p,q):
    '''
    Purpose:
        Building ARMA(p.q) model with statsmodels package for
    comparison purposes

    Inputs:
        df:     dataframe, data
        stock:  string, name of the stock to be analyzed
        year:   integer, start of the decade to be analyzed
        p:      integer, AR parameter
        q:      integer, MA parameter

    Return values:
        scores:       list, AIC and BIC scores
        params:       list, m, phi, theta and sigma parameters
        std_errors:   list, std_errors of parameters
    '''
    #time-series to be modeled
    vY = df.loc[str(year):str(year+10),stock]

    #building model
    model  = ARMA(vY.values,(p,q)).fit(disp=False,trend='c')
    print('Parameters are estimated with package for '+stock+' for
    decade '+ str(year) + ' with p: '+str(p)+' and q: ' + str(q)+'\
    n')

    #parameter estimates
    phis = model.params[1:p+1]
    m = (1-np.sum(phis))*model.params[0]
    thetas = model.params[p+1:p+q+1]
    sigma = np.sqrt(model.sigma2)
    params = [m,phis,thetas,sigma]

    #scores
    scores = [model.aic,model.bic]

    #std. errors
    std_errors = list(np.sqrt(np.diag(model.cov_params())))

    return [scores,params,std_errors]


def interpret_results(df):
    '''
    Purpose:
        Parsing and interpreting results of models into dataframes

    Inputs:
```

```
        df:    dataframe, results of models (scores, params, std.errors)

    Return value:
        df:    dataframe, parsed results
    '''

    df.reset_index(inplace=True)
    df.rename(columns={'level_0':'Stock','level_1':'Decade','level_2':'p','level_3':'q'},inplace=True)

    df[['AIC','BIC']] = pd.DataFrame(df.scores.values.tolist(), index= df.index)
    df['m'] = [i[0] for i in df.params]
    df['m_se'] = [i[0] for i in df.std_errors]


    for j in range(max(df.p)):
        df['phi_'+str(j+1)]= [param[1][j] if p>=(j+1) else np.nan for param,p in zip(df.params,df.p) ]
        df['phi_'+str(j+1)+'_se']= [std_errors[j+1] if p>=(j+1) else np.nan for std_errors,p in zip(df.std_errors,df.p) ]

    for j in range(max(df.q)):
        df['theta_'+str(j+1)]= [param[2][j] if q>=(j+1) else np.nan for param,q in zip(df.params,df.q)]
        df['theta_'+str(j+1)+'_se']= [std_errors[j+p+1] if q>=(j+1) else np.nan for std_errors,p,q in zip(df.std_errors,df.p,df.q) ]

    df['sigma_squared'] = [pow(i[3],2) for i in df.params]
    df['sigma_squared_se'] = [i[-1] for i in df.std_errors]


    df.drop(columns=['scores','params','std_errors'],inplace=True)

    return df



### main
def main():
    #only Apple stock volume will be investigated
    stocks = ['AAPL']

    filename = 'volumes'

    ##read and clean data
    df = read_clean_data(stocks,filename)

    ##plot and extract main descriptives
    print(plot_summarize_data(df))

    #data and model input combinations
    stocks = list(df.columns)
    years = [1990,2000,2010]

```

```python
    #list of possible p and q values
    p_list=[0,1]
    q_list=[0,1]

    #output dfs
    results  = pd.DataFrame(index = pd.MultiIndex.from_product([stocks,years,p_list,q_list]),columns=['scores','params','std_errors'])
    results_package  = copy.deepcopy(results)

    stock = stocks[0]

    #ARMA estimates by computation and package
    for year in years:
        #initial parameters are m=1 and sigma=1
        vP = [np.ones(1),[],[],np.ones(1)*5]
        ##ARMA by computation
        results.loc[stock,year,0,0],vP_noise = ARMA_compute(df,stock,year,0,0,vP)
        results.loc[stock,year,0,1],vP_01 = ARMA_compute(df,stock,year,0,1,vP_noise)
        results.loc[stock,year,1,0],vP_10 = ARMA_compute(df,stock,year,1,0,vP_noise)
        results.loc[stock,year,1,1],vP_11 = ARMA_compute(df,stock,year,1,1,vP_01)

        ##ARMA by packages
        results_package.loc[stock,year,0,0] = ARMA_package(df,stock,year,0,0)
        results_package.loc[stock,year,0,1] = ARMA_package(df,stock,year,0,1)
        results_package.loc[stock,year,1,0] = ARMA_package(df,stock,year,1,0)
        results_package.loc[stock,year,1,1] = ARMA_package(df,stock,year,1,1)

    #parse scores, parameters and std. errors into dataframe
    results_final = interpret_results(results)
    results_package_final = interpret_results(results_package)

    #output results
    results_package_final.to_csv("package.csv")
    results_final.to_csv("computation_robust.csv")

### start main
if __name__ == "__main__":
    main()
```