A Hierarchical Model for Market Beta

Dian Jiao (dj2526), Jiayi He (jh3998), Yifeng Yang (yy2895)

Dec 2019

Contributions

Dian Jiao	Derivation for the model Codes for the evaluation The final report The poster for the project
Jiayi He	Codes for the data processing Codes for the beta estimation The final report
Yifeng Yang	Data collection

1 Background Information

The estimation for market-beta has been crucial for market participants. It is the first composite for nearly all the factor models and the primary measure of how individual assets contribute risk to the market portfolio. It informs investors how tilting assets in and out of market portfolios changes their risk exposures. Thus, it is meaningful in practice for the most primary genre of cross-hedging using the index futures.

Throughout history, many methodologies to estimate the market beta have been proposed by researchers. The most relevant two are the market beta in the CAPM model proposed by Black, Fischer and Jensen [1] and the Vasicek beta proposed by Vasicek and Oldrich A [2]. Using the knowledge in Bayesian Statistics [3], we construct the hierarchical model based on market sectors to estimate the market beta.

1.1 The CAPM Model and OLS Beta

The CAPM is a model for pricing an individual security or portfolio. For individual securities, we are interested in the reward-to-risk ratio β_i shown in the following equation.

$$E(R_i) = R_f + \beta_i(E(R_m) - R_f) \tag{1.1}$$

Because β_i is unobservable, the easiest way to estimate is to regress the excess return of an individual security on the excess market return and get the corresponding reward-to-risk ratio $\hat{\beta}_i$. Specifically, the OLS beta $bols_i$ is calculated from the following regression.

$$r_t = \alpha + \beta r_t^m + \epsilon_t, t = 1, 2, ..., T \tag{1.2}$$

where $\{r_t\}$ and $\{r_t^m\}$ are the return on a security and on a market index.

1.2 The Vasicek Market Beta Estimator

The Vasicek estimator can be viewed either as a Bayesian shrinkage estimator or as the random-effects panel estimator. It requires first computing the OLS market-beta and standard error for each stock within the desired unit of time. Then it requires calculating cross-sectional statistics overall stocks to get the posterior estimation. The details to calculate Vasicek beta are to be discussed in subsection 3.4.

2 Hierarchical Model for Market Beta

2.1 Model Overview

We apply the hierarchical model to CAPM by categorizing all the securities into different sectors.

$$r_t^{i,j} = \alpha_{i,j} + \beta_{i,j} r_t^m + \epsilon_t \tag{2.1}$$

where $t = 1, ..., T$; $i = 1, ..., m$ (for m sectors); $j = 1, ..., n_i$ (for n_i securities within sector i)

For a single stock $S_{i,j}$,

$$\beta_{i,j} \sim N(b_{i,j}, \sigma_{i,j}^2),\ \forall i, j$$

Within each sector i,

$$b_{i,j} \sim N(b_i, \sigma_i^2),\ \forall i, j$$

Between sectors,

$$b_i \sim N(1, \sigma^2),\ \forall i$$

Here $\mathbb{E}(b_i) = 1$ is based on our knowledge that the market beta should be 1.

2.2 The Posterior Distribution

Suppose we have a sample of beta $\hat{\beta}_{i,j}$ for $i = 1, ..., m$, $j = 1, ..., n_i$. Then we can calculate the posterior distributions for b_i and b_{ij} as follows.

The full conditional distribution of b_i is

$$p(b_i|\hat{\beta}_{i1}, ..., \hat{\beta}_{in_i}) \propto \exp(-\tfrac{1}{2\sigma_i^2} \textstyle\sum_j (\hat{\beta}_{ij} - b_i)^2) \exp(-\tfrac{1}{2\sigma^2}(b_i - 1)^2)$$

Thus, the posterior distribution of $b_i \sim N(\hat{\mu}_i, \hat{\sigma}_i^2)$, where

$$\begin{cases} \hat{\mu}_i = \dfrac{1/\sigma^2 + \sum_{j=1}^{n_i} \hat{\beta}_{ij}/\sigma_i^2}{1/\sigma^2 + n_i/\sigma_i^2} \\ \hat{\sigma}_i^2 = \dfrac{1}{1/\sigma^2 + n_i/\sigma_i^2} \end{cases} \tag{2.2}$$

The full conditional distribution of b_{ij} is

$$p(b_{ij}|\hat{\beta}_{ij}) \propto \exp(-\tfrac{1}{2\sigma_{ij}^2}(\hat{\beta}_{ij} - b_{ij})^2) \exp(-\tfrac{1}{2\sigma_i^2}(b_{ij} - b_i)^2)$$

Thus, the posterior distribution of $b_{ij} \sim N(\hat{b}_{ij}, \hat{\sigma}_{ij}^2)$, where

$$\begin{cases} \hat{b}_{ij} = \dfrac{b_i/\sigma_i^2 + \hat{\beta}_{ij}/\sigma_{ij}^2}{1/\sigma_i^2 + 1/\sigma_{ij}^2} \\ \hat{\sigma}_{ij} = \dfrac{1}{1/\sigma_i^2 + 1/\sigma_{ij}^2} \end{cases} \tag{2.3}$$

3 Implementation Details

3.1 Data Description

The data set we use to conduct the numerical tests contains the daily closing price of 3530 composite stocks of Chinese A-shares from Jan 4, 2005 to Jun 29, 2018, based on which we can calculate the daily stock return. The market return we use is the Shanghai Shenzhen CSI 300 Index.

We classified these stocks into 28 categories according to their sectors. The detailed information about these 28 sectors and the number of composite stocks in each sector could be found in the Appendix.

3.2 Posterior Estimation

To implement the forth mentioned Bayesian analysis, we follow the procedure introduced in this part to conduct the numerical tests. First, for each year, we run the OLS regression to get $\hat{\beta}_{i,j}$ and their respective standard errors $s_{i,j}$, for $i = 1, ..., m$, $j = 1, ..., n_i$. These OLS estimations are used as the samples for betas.

We have $(\hat{\beta}_{i,j} - \beta_{i,j})/s_{i,j} \sim t_{T-2}$, and $(\hat{\beta}_{i,j} - \beta_{i,j})/s_{i,j} \sim N(0,1)$ asymptotically for large T. Thus, we use the following equations to estimate σ^2, σ_i^2 and σ_{ij}^2.

$$\begin{cases} \sigma_{ij}^2 = s_{ij}^2 \\ \sigma_i^2 = \dfrac{1}{n_i}\displaystyle\sum_{j=1}^{n_i} s_{ij}^2 \\ \sigma^2 = \dfrac{1}{\sum_{i=1}^{m} n_i}\displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n_i} s_{ij}^2 \end{cases} \tag{3.1}$$

Based on the 3.1, the posterior estimations for $\hat{b}_{ij}$ and $\hat{\sigma}_{ij}$ are

$$\begin{cases} \hat{b}_{ij} = \dfrac{\hat{\mu}_i/\hat{\sigma}_i^2 + \hat{\beta}_{ij}/\sigma_{ij}^2}{1/\hat{\sigma}_i^2 + 1/\sigma_{ij}^2} \\ \hat{\sigma}_{ij} = \dfrac{1}{1/\hat{\sigma}_i^2 + 1/\sigma_{ij}^2} \end{cases} \tag{3.2}$$

where $\hat{\mu}_i$ and $\hat{\sigma}_i^2$ are defined as in 2.2. $\hat{b}_{ij}$ is used as the posterior estimation for β_{ij} in the hierarchical model.

3.3 Benchmark Estimations

To evaluate the performance of our hierarchical model, we use the simple OLS beta and the Vasicek beta. Vasicek beta is also an estimation using Bayesian approach but without hierarchical structure. For each stock i in sector j, the Vasicek estimation is then

$$\tilde{\beta}_{ij} = \frac{\hat{\mu}/\hat{\sigma}_i^2 + \hat{\beta}_{ij}/\sigma_{ij}^2}{1/\hat{\sigma}_i^2 + 1/\sigma_{ij}^2}$$

where $\hat{\mu}$ is calculated as

$$\hat{\mu} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} \hat{\beta}_{ij}$$

3.4 Evaluation for the Estimations

As have been discussed, one important usage of the estimated beta is to do the cross hedging. Thus, to evaluate the estimations, one appropriate perspective is to examine the predictive power of the estimator upon the OLS beta of next year, i.e. the regression models as follows

$$\begin{aligned} \hat{\beta}_t^{OLS} &= a^H + b^H \hat{\beta}_{t-1}^H + \varepsilon_{t-1}^H \\ \hat{\beta}_t^{OLS} &= a^{VCK} + b^{VCK} \hat{\beta}_{t-1}^{VCK} + \varepsilon_{t-1}^{VCK} \\ \hat{\beta}_t^{OLS} &= a^{OLS} + b^{OLS} \hat{\beta}_{t-1}^{OLS} + \varepsilon_{t-1}^{OLS} \end{aligned} \tag{3.3}$$

where $\hat{\beta}^H, \hat{\beta}^{VCK}, \hat{\beta}^{OLS}$ are respectively the betas estimated from hierarchical model, Vasicek approach and OLS regression. We use the R-squared and RMSE as the metrics to evaluate the performance of estimation.

4 Numerical Results

In this part, we present the numerical results using the data and methodologies introduced in section 3.

4.1 Posterior Estimation of $\hat{b}_i$

In the first place, we show the posterior estimate for $\hat{b}_i$'s, which could be viewed as the posterior estimate for the beta of each sector.

As shown in Figure 1, most of the $\hat{b}_i$'s of each sector are strongly correlated across the historical period and fluctuate around 1. This result makes intuitive sense, in that they are all driven by the market as a whole, which has a β of 1.

Of all the sectors, some are sensitive to the market fluctuations, and some are relatively stable across time, e.g. utilities and agriculture. These are stable because they correspond to the basic needs of people.

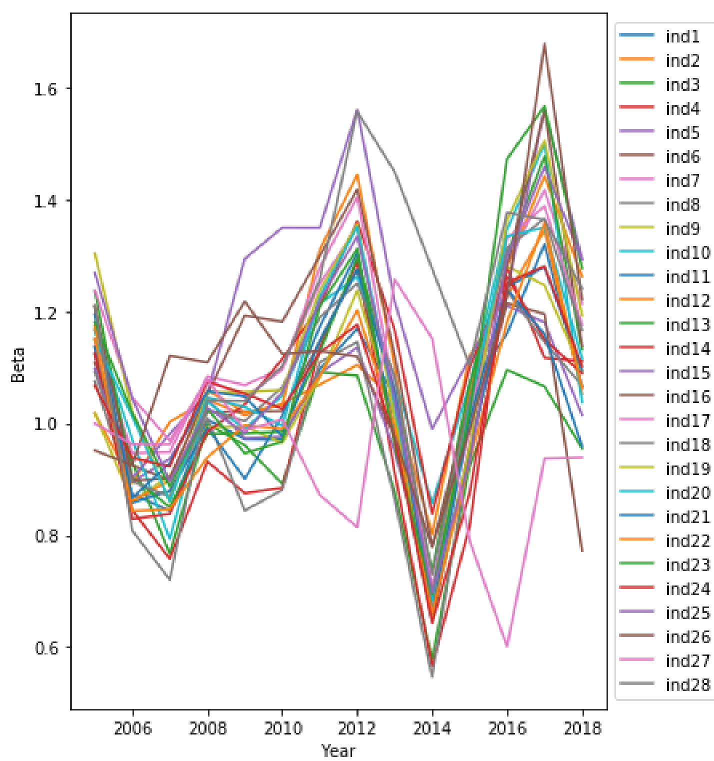

Figure 1: Posterior Estimation of $\hat{b}_i$

4.2 Predictive Performance: Figures

Based on the evaluation regression introduced in subsection 3.4, we could compute the R^2 and RMSE for the regression in each sector.

From Figure 2, we could see that, in terms of R^2, the advantage of the estimation from the hierarchical model we build varies in sectors. In some sectors, e.g., Chemicals, IT, etc., the R^2's from the regression using hierarchical estimates dominate. In others, however, it might be dominated by Vasicek or OLS estimation.

From Figure 3, the performance of hierarchical estimation, the measure by RMSE, dominates the other benchmark estimations in almost all industries.

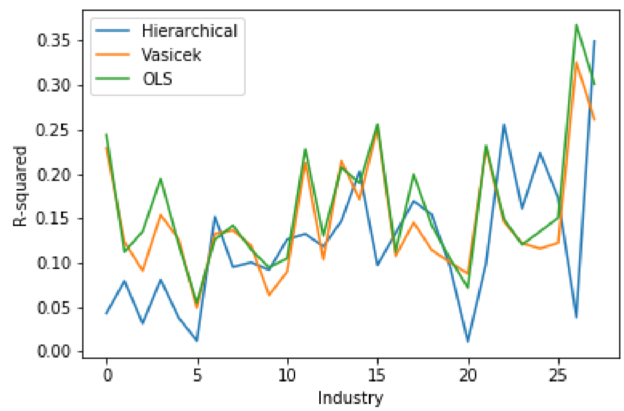

Figure 2: R-Squared of the Evaluation Regression

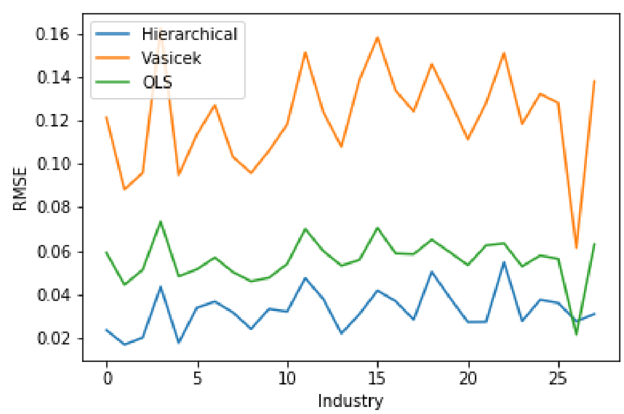

Figure 3: RMSE of the Evaluation Regression

4.3 Predictive Performance: Statistics

Table 1: Evaluation Regressions Predicting One-Year Ahead OLS Market-Beta

Estimation	RMSE	Intercept	Slope	R-Squared
Hierarchical	0.03264	0.72195	0.32428	0.12198
Vasicek	0.12172	0.65406	0.38311	0.14808
OLS	0.05583	0.64029	0.39689	0.16227

The statistics of the regressions are summarized in Table 1. The statistics are the average of the regressions in different sectors.

From the table, we could see that the hierarchical prediction has the smallest RMSE while the largest R^2. However, the difference in R^2 is not significantly large. This shows us that the hierarchical prediction indeed enhances the estimation performance.

Moreover, the intercepts and the slopes of the evaluation regressions are very close. This indicates that the three different estimations are not significantly different from each other, and the estimation does not change wildly after the Bayesian adjustment.

5 Conclusions

5.1 Motivation

Three of us in the group are all majored in Financial Engineering at Columbia. We are very interested in financial topics. The market beta estimation is fundamental to asset pricing theories, and it is closely related to statistics. Our group member Dian Jiao once read about Vasicek's work for the Bayesian approach to estimate market beta and found there are ways to extend it.

Through this project, we enhanced our understanding of the hierarchical model by deriving the model in a different setting and implementing it in a real problem. The model, although normal in nature, is not completely the same as what has been taught in the lectures (e.g., the market beta at the highest level is set 1 one based on our prior knowledge). We also learned how to adjust the model to the settings we are dealing with.

5.2 Conclusions

In this project, we extended the work of Vasicek and applied the hierarchical model in the equity market to estimate the market beta of each stock.

Based on the derivation of the model, we conducted numerical tests using 13 years of stock price data from the Chinese A-share market. The tests include the estimation of OLS, Vasicek, and hierarchical betas and the evaluations of the estimations.

According to the numerical tests, the hierarchical model enhances the estimations in terms of the evaluation regression RMSE. It also enhances the estimation in particular sectors, as measured by the R^2 of the evaluation regression.

5.3 Outlook

There are also perceived aspects to extend the work in this project, which are not covered due to either time/access constraints or that they are beyond the scope of the research.

- The model could be tested on more stock market data, e.g. the U.S. stock market, the full data set of which we did not have access.
- Some of the data we use in this research are not clean due to some missing values, and we drop them in the part of the numerical tests. More careful techniques could be used to enhance the quality of the data.
- More approaches to segment the market and to construct the hierarchy could be used, e.g. the segmentation based on market capitalization, values, etc.

Appendix: Sector Description

Table 2: Sectors and the Number of Composite Stocks

	Sector	Composite Stocks
1	Transportation	109
2	Entertainments	33
3	Media	138
4	Utilities	147
5	Agriculture	94
6	Chemicals	322
7	Medical	277
8	Commercial Trade	97
9	Military Enterprise	49
10	Household Appliance	63
11	Building Material	76
12	Architectural Ornaments	122
13	Real Estate	130
14	Mining	62
15	Non-ferrous Metal	120
16	Machinery	326
17	Automobiles	168
18	Electronic Equipment	222
19	Electrical Equipment	195
20	Clothing	91
21	General	53
22	IT	201
23	Household Manufacturing	124
24	Communication	102
25	Metal	33
26	Bank	26
27	Non-bank Financials	59
28	Food & Beverage	91

Appendix: Data Set

Data set is not available online. The first lines of the data set is shown as in Figure 4.

In [134]:
```
price_df = pd.DataFrame(closeprice)
price_df.head(5)
```

Out[134]:

	0	1	2	3	4	5	6	7	8	9	...	3520	3521	3522	352
0	163.10	150.85	20.94	11.96	18.08	12.42	17.68	9.62	11.83	7.84	...	NaN	NaN	NaN	Na
1	161.60	156.29	21.38	12.24	18.34	12.86	18.15	9.79	12.31	8.11	...	NaN	NaN	NaN	Na
2	163.10	155.43	21.48	12.19	18.25	12.79	18.15	9.73	12.44	8.01	...	NaN	NaN	NaN	Na
3	162.85	156.86	21.38	12.81	20.06	13.29	18.65	9.93	12.55	8.16	...	NaN	NaN	NaN	Na
4	164.85	156.01	21.51	13.20	21.05	14.64	19.58	10.19	13.06	8.26	...	NaN	NaN	NaN	Na

5 rows × 3530 columns

Figure 4: Part of the Data Set

Appendix: Codes

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import h5py
import os
os.chdir('/Users/jiayihe/desktop/Columbia MFE/5224')
import scipy.io as scio
import math

from tabulate import tabulate
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties

filename1 = 'project/closeprice.mat'
data_closeprice = scio.loadmat(filename1)
filename2 = 'project/sector.mat'
data_sector = scio.loadmat(filename2)
filename3 = 'project/date.mat'
data_date = scio.loadmat(filename3)



##--------------------------------Data Processing-------------------------------##

#get different sectors
sector = data_sector['sector'][0][0]
```

```
col_name = []
for i in range(28):
    name = 'ind' + str(i+1)
    col_name.append(name)
ind_dic = {}
ind_dic['ind1'] = list(set(list(sector[0][0])))
sector2 = sector[1][0]
for i in range(1,28):
    ind_dic[col_name[i]] = list(set(list(sector2[i-1][0])))

#get individuaul stock return table
closeprice = data_closeprice['closeprice']
closeprice = closeprice.T
price_df = pd.DataFrame(closeprice)
date = data_date['tdate']
price = np.hstack((date,closeprice))
close_df = pd.DataFrame(price)
ret = np.log(close_df/close_df.shift(1))
ret.drop([0],axis =1,inplace =True)
ret['date'] = date
ret.drop([0],inplace = True)

#get benchmark return table
mkt_ret = pd.read_csv('project/szzz.csv')
mkt_ret = mkt_ret[['Date','Ret']]
mkt_ret.columns = ['date','Ret']
mkt_ret['date'] = pd.to_datetime(mkt_ret['date'])
f = lambda x: int(x.year * 1e4 + x.month * 1e2 + x.day)
mkt_ret['date'] = mkt_ret['date'].apply(f)
df_ret = pd.merge(ret,mkt_ret,how='left',on = 'date')
df_ret['Ret'] = df_ret['Ret'].fillna(0)


##--------------------------------Calculate Prior--------------------------------##

def find_dates(year, date_full):
    ind = np.logical_and((date_full - year * 10e3) > 0, (date_full - year * 10e3) < 10e3)
    return ind
indicator = find_dates(2005,df_ret.date)
year_ret = df_ret[indicator]
test = col_name[0]
ind_li = ind_dic[test]
sec1 = year_ret[ind_li]
mkt1 = year_ret['Ret']


#get beta ij,sigma ij
years = range(2005, 2019)
for sec in col_name:
    beta_dic = {}
    filename_beta = 'project/result/prior/beta/beta_'+sec+'.csv'
    std_dic = {}
    filename_std = 'project/result/prior/std/std_'+sec+'.csv'
```

```
    for year in years:
        indicator = find_dates(year,df_ret.date)
        year_ret = df_ret[indicator]
        ind_li = ind_dic[sec]
        sec1 = year_ret[ind_li]
        mkt1 = year_ret['Ret']
        for company in ind_dic[sec]:
            print('doing', year,company)
            Y = sec1[company]
            X = mkt1
            X = sm.add_constant(X)
            if Y.isna().sum()>20 or (Y==0).sum()>20:
                beta = np.nan
                std = np.nan
            else:
                try:
                    model = sm.OLS(Y,X)
                    results = model.fit()
                    beta = results.params[1]
                    std = results.bse[1]
                except:
                    beta = np.nan
                    std = np.nan
            if company not in beta_dic:
                beta_dic[company] = [beta]
                std_dic[company] = [std]
            else:
                beta_dic[company].append(beta)
                std_dic[company].append(std)
    beta_table = pd.DataFrame(beta_dic)
    std_table = pd.DataFrame(std_dic)
    beta_table.to_csv(filename_beta)
    std_table.to_csv(filename_std)

#get sigma i
file_path = 'project/result/prior'
sigma_i = {}
for sec in col_name:
    df_std = pd.read_csv(file_path+'/std/std_'+sec+'.csv')
    np_std = np.array(df_std)
    for year in range(14):
        np_std_year = np_std[year][1:]
        num_sec = np_std_year.shape[0]-np.isnan(np_std_year).sum()
        where_are_nan = np.isnan(np_std_year)
        np_std_year[where_are_nan] = 0
        sigma = np.power(np_std_year,2).sum()/num_sec

        if sec not in sigma_i:
            sigma_i[sec] = [sigma]
        else:
            sigma_i[sec].append(sigma)
sigma_i_df = pd.DataFrame(sigma_i)

```

```
#get sigma
sec_num = 28
np_sigma_i = np.array(sigma_i_df)
sigma = []

for year in range(14):
    sigma_i_year= np_sigma_i[year][1:]
    sigma_year = sigma_i_year.sum()/sec_num
    sigma.append(sigma_year)

##--------------------------------Calculate Posterior--------------------------------##
#get sector posterior
sigma_path = 'project/result/prior'
mu_i = {}
for sec in col_name:
    df_beta = pd.read_csv(file_path + '/beta/beta_' + sec + '.csv', index_col=0)
    for year in range(14):

        df_beta_year = df_beta.iloc[year]
        sum_beta = df_beta_year.sum()
        num_beta = df_beta_year.shape[0] - df_beta_year.isna().sum()

        sigma_year = sigma[year]
        sigma_i_year = sigma_i_df[sec][year]

        mu = (1 / sigma_year + sum_beta / sigma_i_year) / (1 / sigma_year + num_beta /
    sigma_i_year)

        if sec not in mu_i:
            mu_i[sec] = [mu]
        else:
            mu_i[sec].append(mu)
mu_i_df = pd.DataFrame(mu_i)

sigma_path = 'project/result/prior'
sigma_i_hat = {}
for sec in col_name:
    df_beta = pd.read_csv(file_path + '/beta/beta_' + sec + '.csv', index_col=0)
    for year in range(14):

        df_beta_year = df_beta.iloc[year]
        # sum_beta = df_beta_year.sum()
        num_beta = df_beta_year.shape[0] - df_beta_year.isna().sum()

        sigma_year = sigma[year]
        sigma_i_year = sigma_i_df[sec][year]

        sigma_hat = 1 / (1 / sigma_year + num_beta / sigma_i_year)

        if sec not in sigma_i_hat:
            sigma_i_hat[sec] = [sigma_hat]
        else:
            sigma_i_hat[sec].append(sigma_hat)
```

```
sigma_i_hat_df = pd.DataFrame(sigma_i_hat)


#get single stock posterior
sigma_path = 'project/result/prior/std/'
beta_path = 'project/result/prior/beta/'
posterior_path_beta = 'project/result/posterior/b_ij/'
posterior_path_sigma = 'project/result/posterior/sigma_ij/'

years = range(14)
for sec in col_name:
    b_dic = {}
    sigma_dic = {}
    filename_b = posterior_path_beta+'b_'+sec+'.csv'
    filename_sigma = posterior_path_sigma+'sigma_'+sec+'.csv'
    for year in years:
        mu_i = mu_i_df[sec][year]
        sigma_i = sigma_i_hat_df[sec][year]
        beta_df = pd.read_csv(beta_path+'beta_'+sec+'.csv',index_col = 0)
        sigma_df = pd.read_csv(sigma_path+'std_'+sec+'.csv',index_col = 0)
        for company in ind_dic[sec]:
            print('doing', year,company)
            beta_ij = beta_df[str(company)][year]
            std_ij = sigma_df[str(company)][year]
            try:
                b_ij = (mu_i/sigma_i+beta_ij/std_ij)/(1/sigma_i+1/std_ij)
            except:
                b_ij = np.nan
            try:
                sigma_ij = 1/(1/sigma_i+1/std_ij)
            except:
                sigma_ij = np.nan
            if company not in b_dic:
                b_dic[company] = [b_ij]
                sigma_dic[company] = [sigma_ij]
            else:
                b_dic[company].append(b_ij)
                sigma_dic[company].append(sigma_ij)
    b_table = pd.DataFrame(b_dic)
    sigma_table = pd.DataFrame(sigma_dic)
    b_table.to_csv(filename_b)
    sigma_table.to_csv(filename_sigma)

##-------------------------------Access Previous Results------------------------------##
n_ind = 28

dta = dict()
for ind in range(n_ind):
    df = pd.read_csv('project/result/posterior/b_ind' + str(ind + 1) + '.csv', index_col =
    0)
    dta_ = np.empty((0, 2))
    for col in df.columns:
```

```python
        df_ = pd.concat((df[col][:-1], df[col][1:].reset_index().drop('index', axis = 1)), axis = 1).dropna()
        dta_ = np.vstack((dta_, df_.values))
    dta[ind] = dta_

dta_ols = dict()
for ind in range(n_ind):
    df = pd.read_csv('project/result/prior/beta/beta_ind' + str(ind + 1) + '.csv', index_col = 0)
    dta_ = np.empty((0, 2))
    for col in df.columns:
        df_ = pd.concat((df[col][:-1], df[col][1:].reset_index().drop('index', axis = 1)), axis = 1).dropna()
        dta_ = np.vstack((dta_, df_.values))
    dta_ols[ind] = dta_

df_std = pd.DataFrame()
df_avg = pd.DataFrame()
for ind in range(n_ind):
    df1 = pd.read_csv('project/result/prior/std/std_ind' + str(ind + 1) + '.csv', index_col = 0)
    df2 = pd.read_csv('project/result/prior/beta/beta_ind' + str(ind + 1) + '.csv', index_col = 0)
    df_std = pd.concat((df_std, df1), axis = 1)
    df_avg = pd.concat((df_avg, df2), axis = 1)
sg = df_std.mean(axis = 1)
mu = df_avg.mean(axis = 1)

##--------------------------------Calculate Vasicek Beta--------------------------------##
dta_vck = dict()
for ind in range(n_ind):
    df1 = pd.read_csv('project/result/prior/std/std_ind' + str(ind + 1) + '.csv', index_col = 0)
    df2 = pd.read_csv('project/result/prior/beta/beta_ind' + str(ind + 1) + '.csv', index_col = 0)

    sg1 = pd.DataFrame([sg for i in range(df1.shape[1])]).T
    sg1.columns = df1.columns
    mu1 = pd.DataFrame([mu for i in range(df1.shape[1])]).T
    mu1.columns = df1.columns

    b_vck = df2 * sg1 / (sg1 + df1) + mu1 * df1 / (sg1 + df1)
    b_vck.to_csv('project/result/vck/vck_ind' + str(ind + 1) + '.csv')

    dta_ = np.empty((0, 2))
    for col in b_vck.columns:
        df_ = pd.concat((b_vck[col][:-1], b_vck[col][1:].reset_index().drop('index', axis = 1)), axis = 1).dropna()
        dta_ = np.vstack((dta_, df_.values))
    dta_vck[ind] = dta_

##--------------------------------Evaluation Regression--------------------------------##
r2 = np.empty((0, 3))
```

```
rmse = np.empty((0, 3))
intercept = np.empty((0, 3))
slope = np.empty((0, 3))
for ind in range(n_ind):
    X = np.empty(shape=dta[ind].shape, dtype=np.float)
    X[:, 0] = 1
    X[:, 1] = dta[ind][:, 0]
    Y = dta[ind][:, 1]

    model = sm.OLS(Y, X)
    result = model.fit()

    X_ols = np.empty(shape=dta_ols[ind].shape, dtype=np.float)
    X_ols[:, 0] = 1
    X_ols[:, 1] = dta_ols[ind][:, 0]
    Y_ols = dta_ols[ind][:, 1]

    model_ols = sm.OLS(Y_ols, X_ols)
    result_ols = model_ols.fit()

    X_vck = np.empty(shape=dta_vck[ind].shape, dtype=np.float)
    X_vck[:, 0] = 1
    X_vck[:, 1] = dta_vck[ind][:, 0]
    Y_vck = dta_vck[ind][:, 1]

    model_vck = sm.OLS(Y_vck, X_vck)
    result_vck = model_vck.fit()

    r2 = np.vstack((r2, [result.rsquared, result_ols.rsquared, result_vck.rsquared]))
    rmse = np.vstack((rmse, [result.mse_resid, result_ols.mse_resid, result_vck.mse_resid])
    )
    intercept = np.vstack((intercept, [result.params[0], result_ols.params[0], result_vck.
    params[0]]))
    slope = np.vstack((slope, [result.params[1], result_ols.params[1], result_vck.params
    [1]]))

##--------------------------------Visualization-------------------------------##
b = pd.read_csv('project/result/posterior/b_i.csv', index_col = 0)
fig, ax = plt.subplots(figsize = (6, 8))
ax.plot(range(2005, 2019), b)
ax.legend(b.columns, loc='center left', bbox_to_anchor=(1, 0.5))
ax.set_xlabel('Year')
ax.set_ylabel('Beta')
fig.savefig('b.png')

plt.plot(r2)
plt.legend(['Hierarchical', 'Vasicek', 'OLS'])
plt.xlabel('Industry')
plt.ylabel('R-squared')
plt.savefig('rsq.png')

plt.plot(rmse)
plt.legend(['Hierarchical', 'Vasicek', 'OLS'], loc='upper left')
```

```
plt.xlabel('Industry')
plt.ylabel('RMSE')
plt.savefig('rmse.png')

table = [[i, rmse.mean(axis = 0)[i], intercept.mean(axis = 0)[i], slope.mean(axis = 0)[i],
    r2.mean(axis = 0)[i]] for i in range(3)]
headers = ['Estimation', 'RMSE', 'Intercept', 'Slope', 'R-squared']
print(tabulate(table, headers, tablefmt='pipe'))
```

Listing 1: Sources Codes for the Project

References

[1] Fischer Black, Michael C Jensen, Myron Scholes, et al. The capital asset pricing model: Some empirical tests. *Studies in the theory of capital markets*, 81(3):79–121, 1972.

[2] Oldrich A Vasicek. A note on using cross-sectional information in bayesian estimation of security betas. *The Journal of Finance*, 28(5):1233–1239, 1973.

[3] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.