MTH 4600 - Project 5
Principal Component Analysis

Martin Bachurin
Anton Gurshumov
John Hemminger

Baruch College

April 4, 2023

For this project, we subject the 30 components of the Dow Jones Industrial Average to a principle component analysis. We are given a time series of the daily percent price change for each of the 30 stocks for the year 2020 (taken from www.investing.com/equities/united-states) and stored inside the 30 x 253 matrix X. For question 1, we compute the sample covariance and correlation matrices for the data by running the matrix X through the functions "Covariance()" and "Correlation()". Next, we test if any of the pairs identified by the ticker have a correlation in excess of 0.8 or if any of the pairs are negative by running through each row and column. As a technical note, since the correlation matrix is symmetric, we can run the loop so that we do not get any repetitions (this is shown in the double for loop in the code). The following is a list of these results.
Correlation(CAT,DOW) = 0.804226
Correlation(AXP,GS)= 0.850246
Correlation(AXP,HON)= 0.859499
Correlation(GS,HON) = 0.812033
Correlation(AXP,JPM) = 0.890625
Correlation(CVX,JPM) = 0.802199
Correlation(GS,JPM) = 0.890224
Correlation(HON,JPM) = 0.828969
Correlation(APPL,MSFT) = 0.839093
Correlation(AXP,V) = 0.824185
Correlation(HON,V) = 0.821373
There is no negative correlation between the returns on the Dow Jones stocks, which means that the entire matrix is positive. What this means is that there are no inverse relationships between any two of the 30 stocks according to the sample correlation matrix. Also, the reported stock pairs having a high correlation implies that they move closely together in the market.

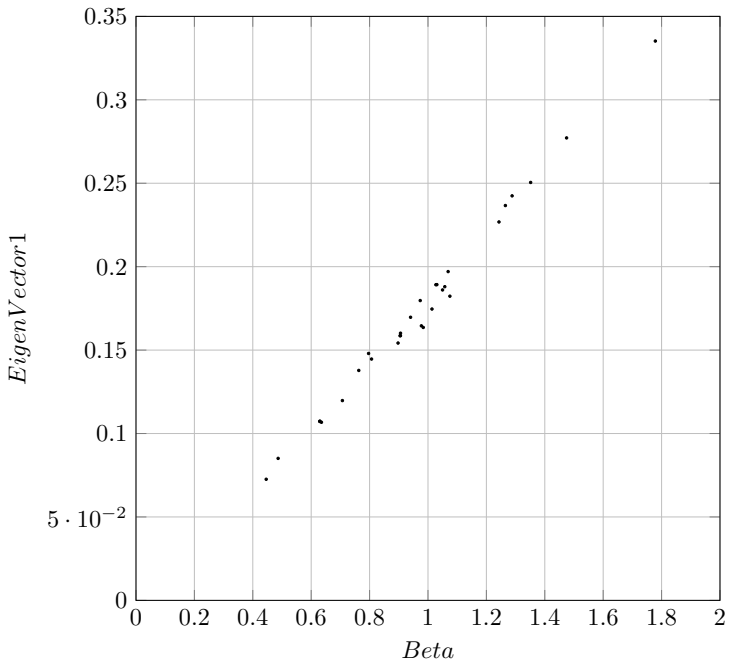
For question 2, we compute the eigenvalues and eigenvectors of the covariance matrix by running the associated functions QRalgorithm(Covariance Matrix) for the eigenvalues and Evector(Covariance Matrix) for the eigenvectors. We see that the first Eigenvector, with eigenvalue = 166.7434, has only positive components, while all the others have negative components.

For question 3, we calculate the Beta's of each stock, which is defined as a scaling factor measuring the stock's expected response to changes in the overall market. It is calculated by

$$\beta = \frac{\mathrm{Cov}(R,M)}{\mathrm{Var}(M)}$$

. Where R is the particular stock's return on any particular day and M is the total Dow's return on a particular day. Next, we pair each stock's Beta with the i^{th} component in the first eigenvector and produce a scatterplot. We did this by creating a text file with each of the 30 pairs of numbers written on each line and then using the package pgfplots to produce a simple scatterplot. This is shown in the figure below.

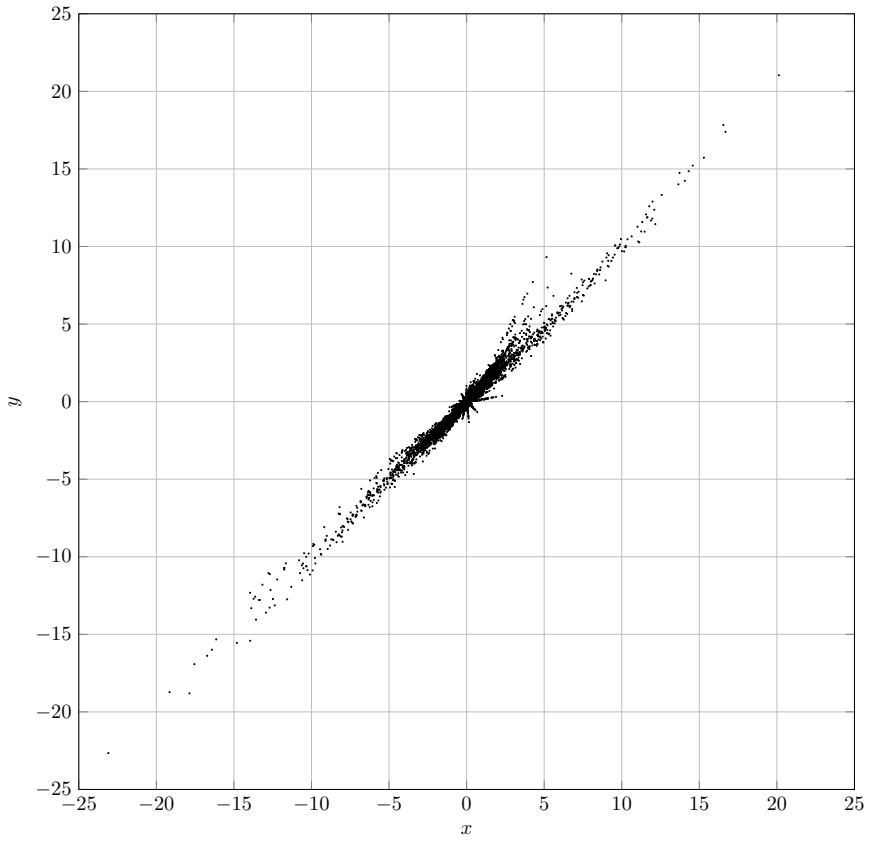
4


0.35
0.3
0.25
0.2
0.15
0.1
$5 \cdot 10^{-2}$
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
$EigenVector1$
$Beta$

As we can see, the higher the component value of the first eigenvector, the higher the Beta value of the stock will be. We also notice that the majority of the stocks have a beta of around 1, and a line could easily be interpolated if one wished with a slope of 0.18 that approximately intercepts the origin. Additionally, this result makes sense since all of the components of the first eigenvector are positive.

For question 4, for each $1 \leq i \leq 30$ and $1 \leq t \leq 253$, we calculated two values, $x = \beta_i M_t$ and $y = Y_{1t} q_i$. M_t is the total DOW return on day t, Y_{1t} is the first principal component of the t^{th} column of X, and q_i is the i^{th} component of the first eigenvector. We calculated all 7590 points of each x and y, by looping through each day, and each stock, multiplying the appropriate values. These values were stored in arrays, then printed as space-separated pairs as before in a text file, which in turn was converted into the following scatter-plot.

6



For question 5, given the above scatter plots, we can give an interpretation of the first eigenvector. The first thing that we notice is the positive linear relationship in both plots. This is expected since the first eigenvector is entirely positive. In the case of the first scatterplot, it shows that the larger the component of the eigenvector is, the higher the associated sensitivity to changing market conditions will be as well. For the second scatterplot, we can say that the data has a slope of 1 and that it is also approximately symmetric about the line $y = -x$. This implies that the computed values of x and y are highly correlated with each other. Note that this large eigenvalue accounts for $166.7424/266.393 = 62.59\%$ of the variance of the total Dow returns. Also, the first eigenvector being strictly positive tells us that 62.59% of the variance of total returns of the DJIA can be explained by shifting the mean curve upwards. This is a significant proportion when it comes to estimation! In general, from the first eigenvector being positive, we can say that the DJIA saw an upward shift in 2020 despite the COVID recession. This is most likely caused by the recovery seen throughout the year.

For question 6, we report the eigenvector corresponding to the second largest eigenvalue of 21.6608 with components identified by ticker.
-0.267461 :: AMGN
-0.258947 :: APPL
0.228736 :: AXP
0.538883 :: BA
0.0319141 :: CAT
-0.278956 :: CRM
-0.165412 :: CSCO
0.162556 :: CVX
0.0844698 :: DIS
0.0654437 :: DOW
0.061286 :: GS
-0.139699 :: HD
0.112763 :: HON
-0.0430852 :: IBM
-0.238951 :: INTC
-0.149755 :: JNJ
0.14075 :: JPM
-0.00197232 :: KO
-0.0172021 :: MCD

-0.0298399 :: MMM
-0.147256 :: MRK
-0.28326 :: MSFT
-0.0169963 :: NKE
-0.19645 :: PG
0.0454874 :: TRV
-0.144592 :: UNH
-0.0556552 :: V
-0.0962104 :: VZ
-0.0576609 :: WBA
-0.238394 :: WMT

Of the values above, the ones with an excess of 0.15 in absolute value are:
-0.267461 :: AMGN - Bio-pharmaceutical
-0.258947 :: APPL - Information technology
0.228736 :: AXP - Financial services
0.538883 :: BA - Aerospace and defense
-0.278956 :: CRM - Information technology
-0.165412 :: CSCO - Information technology
0.162556 :: CVX - Petroleum industry
-0.238951 :: INTC - Semiconductor industry
-0.28326 :: MSFT - Information technology
-0.19645 :: PG - Fast-moving consumer goods
-0.238394 :: WMT - Retailing

We can see that the information technology sector is the most influential sector for the second eigenvector. Also, the stocks listed above contribute to most of the total variation accounted for by this eigenvector. We can interpret this eigenvector as adding multiple humps/valleys to the shape of the variation of the total returns from the DJIA. This accounts for $21.6608/266.393 = 8.1\%$ of the total variation. In general, we can say that the year 2020 was very hectic for the IT sector, with significant losses as well as recoveries. This is seen in the absolute value of the components being greater than 0.15.

Our group separated the work relatively uniformly over the interval of time required to complete the assignment. We met on a regular basis and worked through each problem at a time, always listening to criticism and different points of view. We all coded together, and if something was unclear to someone when writing the code, we always explained precisely what was happening line by line. Our group has a great interest in these projects and subject matter; therefore, we have had no issues with scheduling and collaborating as a team. We genuinely enjoy working on these projects together.

Appendix

Listing 1: Code

```
// These functions are defined below.
void AllocateMemory ();
void GetComponentData(void);
void GetM(void);
void Preliminaries ();
void ReportForScatter(double*,double*);
void ReportForScatterBeta(double*,double*);

// Global variables
double *M;
char **ticker;
int n=3041, p=8;
int *mm, *dd, *yy;
double **T, **Xtilde, **X, **D, **Q, **Y, **V, *mu,**L, **Lambda;
double* Beta = new double[31];
double* EVect_1 = new double[31];
#include "Functions.h"
#include <iostream>
int main () {
   double **E;
   int p, n, i;

   // Allocate some array space.
   AllocateMemory ();
//Read in the DJI data. These are daily returns for the 30 components for 2020.
//Put it into a 30 x 253 array X. For your convenience, the data has already been
//shifted so each row has mean zero. The means are very small and this has minimal
//impact.
   GetComponentData();
// Now read in the daily returns of the total Dow average for 2020. This is a list
// of 253 returns and is put into the list M.  This is used in problem 3
// to compute the individual components' betas. Also used in problem 4. The data
// here has also been shifted to have mean zero.
   GetM();
   // Get the dimensions of DJI.
   n = Columns (X);       // Should be 253
   p = Rows (X);          // Should be 30
   V = Covariance (X);
// Compute eigenvalues and store them in E
   E = QRalgorithm (V);

// Compute the eigenvectors.
   Q = Evector (V, E);
   L = Correlation(X);
   Show(L);
```

```
//By using Show L, we see that L is positive.
// No need to check for any negative values.
//Check which stock pairs have correlation greater than 0.8
//L is symmetric 30 x 30 , therefore we loop though the columns while k<j
   for(int j = 1; j <= 30; ++j){
      for (int k = 1; k < j; ++k) {
         if(L[j][k] > 0.8){          //test condition
            std::cout<<ticker[k]<<"\n";
            std::cout<<ticker[j] <<"\n"<<"Has correlation = "<<L[j][k]<<"\n\n";
         }
      }
   }
   Pause();
   // Number 3 after getting M.
   // beta = COV(R,M) / VAR(M)
   // M is the overall market return that same day.
   // R is stock return out of 253 days.
   // compute variance of M that same day, for each stock
   double* var = new double[31];
   double* covar = new double[31];
   for(int s = 1;s<=30; ++s){ // for each stock
      for(int d = 1;d <= 253;++d){ //compute the var, covar for each day.
         var[s] = ((d-1)*var[s] + M[d]*M[d]) / d;
         covar[s] =((d-1)*covar[s] + M[d]* X[s][d]) / d;
      }
   }
// compute beta for each stock, and show the associated ticker.
   for(int b = 1;b<=30;++b){
      Beta[b] = covar[b] / var[b];
      std::cout<<"Beta for stock "<<ticker[b]<<"\n is = "<<Beta[b]<<"\n";
   }
   Pause();
   for(int q = 1;q<=30;++q){
      std::cout<<Q[q][1]<<"\n";
      EVect_1[q] = Q[q][1];
   }
// generate a txt file for (Beta,Eigenvector1) points,
// which will then be run through overleaf for scatterplot
   ReportForScatterBeta(Beta,EVect_1);
   //END of question 3
   double ** QT;
   QT = Transpose(Q);
   // Compute the principal components of each column of X. Put them into the
   //    corresponding column of Y.
   Y = Multiply (QT, X);
   Pause();
//Question 4: Compute x = B_i*M_t ; and y = Y_1t*q_i;
// where Y1t is the first principle component of the t^th column of X
// and q_i is the i^th component of the first eigenvector.
```

```
// create memory to store points.
   double* x = new double[7591]; double* y = new double[7591];

   // Compute x and y. Note: 1 <= i <= 30 and 1 <= t <= 253
   int count = 0;
   for(int t = 1;t <= 253; ++t){ // loop through days.
      for(int i = 1; i<= 30; ++i){  //loop through all betas
         x[count] = M[t]*Beta[i];
         ++count;
      }
   }

   Pause();
   // reset count for the next 7590 points of data for y
   count = 0;
   for(int t = 1;t <= 253; ++t){ // loop through days
      for(int i = 1; i<= 30; ++i ){  // loop through 30 coomponents of Eigenvector 1.
         y[count] =  Y[1][t] * Q[i][1];
         ++count;
      }
   }
// call function to write vector x, y into a txt file.
// This will be converted to TeX.
   ReportForScatter(x,y);

   // Start of Question 6 //
// Show eigenvector 2
   for(int i=1;i<=30;++i){
      std::cout<< Q[i][2]<<" :: "<<ticker[i]<<"\n";
   }
   std::cout<<"\n\n";
// Show components of eigenvector 2 such that the absolute value is greater than 0.15
   for(int i=1;i<=30;++i){
      if(Q[i][2] > 0.15 || Q[i][2] * -1 > 0.15 ){
         std::cout<< Q[i][2]<<" :: "<<ticker[i]<<"\n";
      }
      // Tech companies are heavily represented in this eigenvector
   }
   // END of Question 6 //
   Pause ();
   delete[] x; delete[] y; delete[] var;
   delete[] covar; delete[] Beta; delete[] EVect_1;
}
// This function reads in the 253 daily returns
// for each of the 30 components of the Dow for the year 2020.
void GetComponentData() {
   int i, k, t;
   double R, mu;
```

```
    char input[100];
    FILE *fp;
    fp = fopen ("DJI.txt", "r");
    for (i = 1; i <= 30; i++) {
       // Read in stock i's return data.
       // Get the tickers for each stock.
       fgets (input, 99, fp);
       for (k = 0; k < 10; k++) {
          ticker[i][k] = input[k];
          if (ticker[i][k] == '\n') {
             ticker[i][k] = '\0';
             break;
          }
       }
       // Read in the 253 daily returns in percent.
       mu = 0.0;
       for (t = 1; t <= 253; t++) {
          // Month t return for stock i, put it into X[i][t].
          fgets (input, 99, fp);
          sscanf (input, "%lf", &R);
          X[i][t] = R;
          mu += R / 253.0;
       }
       // Make the mean return 0 (this has minimal impact).
       for (t = 1; t <= 253; t++) {
          X[i][t] -= mu;
       }
    }
    fclose (fp);
    return;
}
// This function writes the coordinates for plotting into a txt file before TeX.
// Note that all arrays are already organized and ordered properly.
// takes in arrays as parameters
void ReportForScatterBeta(double* Beta, double* EVect_1) {
     FILE *plotBeta;                              // file variable plotBeta
     plotBeta = fopen("Scat_Plot_Beta.txt","w"); // open txt file in write mode
     for(int i = 1 ;i <= 30;++i){
         fprintf(plotBeta,"%lf",Beta[i]);      // write first component of array,
         fprintf(plotBeta,"%s"," ");           // separate with space on that line
         fprintf(plotBeta,"%lf\n",EVect_1[i]);
    // write y coordinate point, followed by new line
     }
     fclose(plotBeta);      // close plotBeta file.
}
// This function writes the coordinates for plotting into a txt file before TeX.
// This is the same function as above, except we are doing it for more points.
// Note that all arrays are already organized and ordered properly.
void ReportForScatter(double* x, double* y) {
```

```
      FILE *plot;
      plot = fopen("Scat_Plot_xy3","w");
      for(int i = 1 ;i <= 7590;++i){
           fprintf(plot,"%lf",x[i]);
           fprintf(plot,"%s"," ");
           fprintf(plot,"%lf\n",y[i]);
      }
      fclose(plot);
}
// This function reads in the daily aggregate returns of the Dow Jones Industrial Average
// for the year 2020.
void GetM() {
   int t;
   double R, mu=0.0;
   char input[100];
   FILE *fp;
   fp = fopen ("M.txt", "r");
   for (t = 1; t <= 253; t++) {
      // Month t return for the DJIA (the total Dow).
      fgets (input, 99, fp);
      sscanf (input, "%lf", &R);
      M[t] = R;
      mu += R / 253.0;
   }
   // Make the market returns mean zero (this has minimal impact).
   for (t = 1; t <= 253; t++) {
      M[t] -= mu;
   }
   fclose (fp);
   return;
}
// Allocate some array space.
void AllocateMemory () {
   int i;
   X = Array (30, 253);
   M = List (253);
   ticker = (char **) calloc (31, sizeof (char *));
   for (i = 1; i <= 30; i++) {
      ticker[i] = (char *) calloc (10, sizeof (char));
   }
   return;
}
```