Homework 1

Ze Yang (zey@andrew.cmu.edu)

Due Wednesday, November 1 at 5:30 PM

You should submit the Rmd file for your analysis. Name the file as `YOURANDREWID_HW1.Rmd` and submit it via Canvas. Also submit the `.pdf` file that is produced.

Part 1:

Reconsider the data set that was presented in the final exam for Mini 1. Create a new data frame that gives the day-to-day changes in all of the rates. Run a principal components analysis on these vectors. Is PCA effective in reducing the dimensionality of the rate change vectors? Try it with and without scaling the variables first and describe how the results change.

```r
# load data, the code used in mini1 final
fileheader = read.table("commercial_paper_rates.csv",sep=",",
                        nrows=6,stringsAsFactors=FALSE)
cprfullnames = as.character(c("Time.Period",fileheader[1,-1]))
cprdata = read.table("commercial_paper_rates.csv",sep=",", header=T,
                     skip=5,stringsAsFactors = FALSE,
                     na.strings=c("ND","NA"))
cprdata$Time.Period = as.Date(cprdata$Time.Period, format="%Y-%m-%d")
df = cprdata
names = c('time','nf.1d.AA','nf.7d.AA','nf.15d.AA',
          'nf.30d.AA','nf.60d.AA','nf.90d.AA',
          'nf.1d.A2P2','nf.7d.A2P2','nf.15d.A2P2',
          'nf.30d.A2P2','nf.60d.A2P2','nf.90d.A2P2',
          'f.1d.AA','f.7d.AA','f.15d.AA',
          'f.30d.AA','f.60d.AA','f.90d.AA',
          'asb.1d.AA','asb.7d.AA','asb.15d.AA',
          'asb.30d.AA','asb.60d.AA','asb.90d.AA')
colnames(df) = names
df = df[!is.na(df$time),]
```

```r
# calculate rates shift
df.shift = diff(as.matrix(df[,2:25]))
df.shift = as.data.frame(df.shift)

# calculate rates shift with scaling
df.shift.scale = scale(df.shift[complete.cases(df.shift),])

# pca
pca.1 = princomp(df.shift, subset=complete.cases(df.shift))
pca.2 = princomp(df.shift.scale)
```

```r
summary(pca.1)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4
## Standard deviation     0.2125105 0.1152238 0.10521763 0.08534759
## Proportion of Variance 0.3988022 0.1172415 0.09776283 0.06432492
## Cumulative Proportion  0.3988022 0.5160437 0.61380655 0.67813147
##                            Comp.5     Comp.6     Comp.7     Comp.8
## Standard deviation     0.06999202 0.06349741 0.06111537 0.06007489
## Proportion of Variance 0.04326072 0.03560482 0.03298357 0.03187005
## Cumulative Proportion  0.72139219 0.75699701 0.78998058 0.82185063
##                            Comp.9    Comp.10    Comp.11    Comp.12
## Standard deviation     0.05346768 0.05150504 0.04665964 0.04277101
## Proportion of Variance 0.02524523 0.02342590 0.01922558 0.01615459
## Cumulative Proportion  0.84709586 0.87052176 0.88974734 0.90590193
##                           Comp.13    Comp.14    Comp.15     Comp.16
## Standard deviation     0.03797872 0.03783245 0.03495439 0.033545953
## Proportion of Variance 0.01273730 0.01263938 0.01078947 0.009937497
## Cumulative Proportion  0.91863923 0.93127861 0.94206808 0.952005579
##                            Comp.17     Comp.18     Comp.19     Comp.20
## Standard deviation     0.032606195 0.029661307 0.028966165 0.027525308
## Proportion of Variance 0.009388517 0.007769218 0.007409327 0.006690539
## Cumulative Proportion  0.961394096 0.969163314 0.976572640 0.983263180
##                            Comp.21    Comp.22     Comp.23     Comp.24
## Standard deviation     0.025793501 0.02484151 0.020239544 0.014256505
## Proportion of Variance 0.005875128 0.00544945 0.003617414 0.001794828
## Cumulative Proportion  0.989138307 0.99458776 0.998205172 1.000000000
```

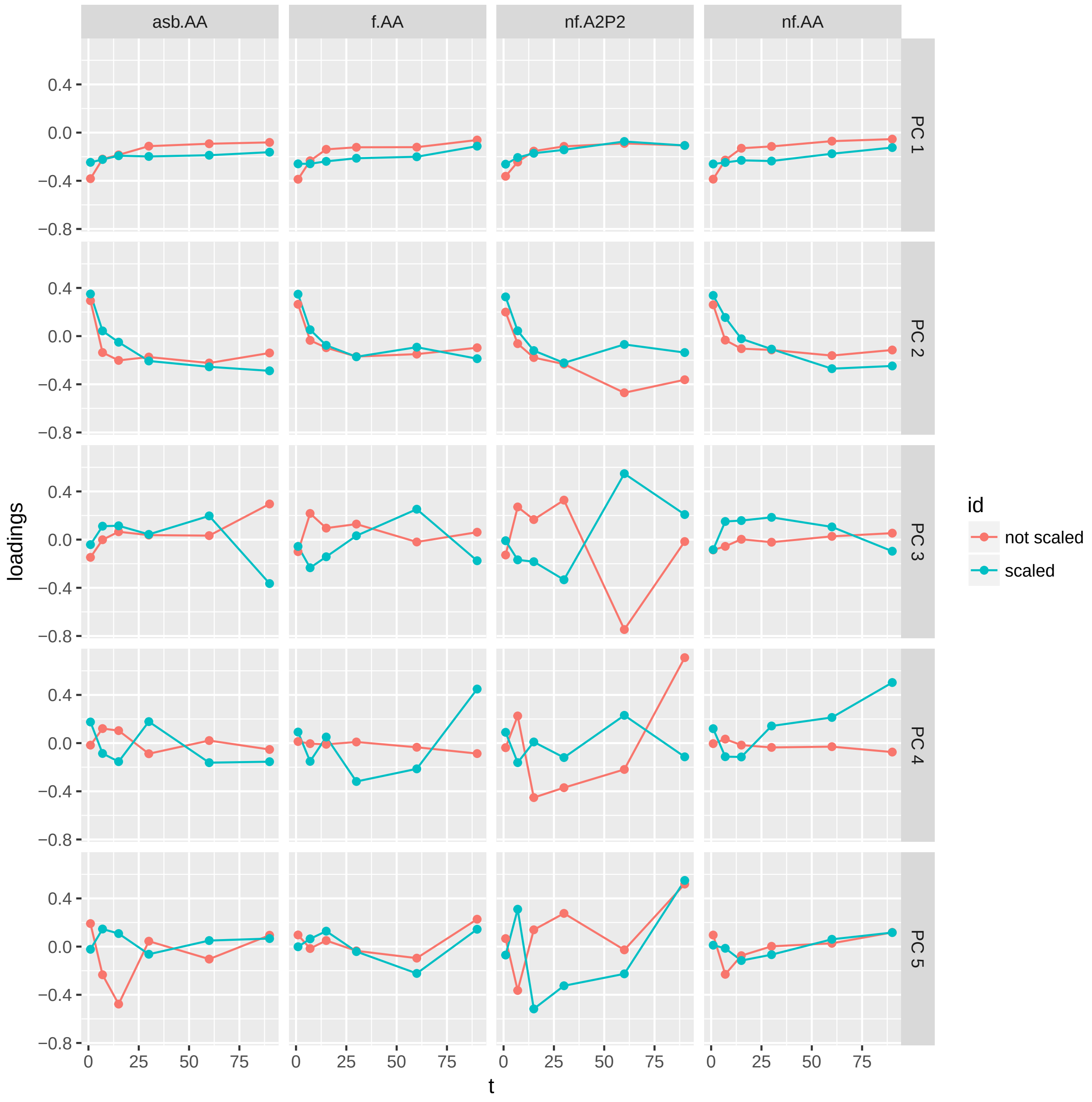
```r
summary(pca.2)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4
## Standard deviation     2.9521963 1.6370103 1.28863462 1.18665513
## Proportion of Variance 0.3635021 0.1117685 0.06925897 0.05873074
## Cumulative Proportion  0.3635021 0.4752705 0.54452949 0.60326023
##                            Comp.5     Comp.6     Comp.7     Comp.8
## Standard deviation     1.04312482 0.92882960 0.89692828 0.88261934
## Proportion of Variance 0.04538256 0.03598227 0.03355304 0.03249102
## Cumulative Proportion  0.64864278 0.68462505 0.71817809 0.75066911
##                            Comp.9    Comp.10    Comp.11    Comp.12
## Standard deviation     0.83600736 0.80521431 0.77747488 0.73769171
## Proportion of Variance 0.02914987 0.02704204 0.02521095 0.02269688
## Cumulative Proportion  0.77981898 0.80686101 0.83207196 0.85476884
##                           Comp.13    Comp.14    Comp.15    Comp.16
## Standard deviation     0.70574170 0.65719387 0.65042812 0.62462156
## Proportion of Variance 0.02077342 0.01801372 0.01764473 0.01627235
## Cumulative Proportion  0.87554226 0.89355598 0.91120072 0.92747307
##                          Comp.17    Comp.18    Comp.19    Comp.20
## Standard deviation     0.6049848 0.58851112 0.56774473 0.53101069
## Proportion of Variance 0.0152653 0.01444527 0.01344382 0.01176042
## Cumulative Proportion  0.9427384 0.95718364 0.97062746 0.98238788
##                           Comp.21     Comp.22     Comp.23     Comp.24
## Standard deviation     0.49601981 0.315201020 0.227520714 0.158498503
## Proportion of Variance 0.01026159 0.004143732 0.002159028 0.001047772
## Cumulative Proportion  0.99264947 0.996793200 0.998952228 1.000000000
```

```
compose.loadings = function(pca, k, identifier) {
  df = data.frame(loadings=pca$loadings[,k])
  df$t = rep(c(1,7,15,30,60,90),4)
  df$type = rep(
    c('nf.AA','nf.A2P2','f.AA','asb.AA'), each=6)
  df$order = rep(paste('PC',k), nrow(df))
  df$id = rep(identifier, nrow(df))
  return(df)
}

# PC1,2,3,4,5 loadings
loadings = rbind(
  compose.loadings(pca.1, 1, 'not scaled'),
  compose.loadings(pca.2, 1, 'scaled'),
  compose.loadings(pca.1, 2, 'not scaled'),
  compose.loadings(pca.2, 2, 'scaled'),
  compose.loadings(pca.1, 3, 'not scaled'),
  compose.loadings(pca.2, 3, 'scaled'),
  compose.loadings(pca.1, 4, 'not scaled'),
  compose.loadings(pca.2, 4, 'scaled'),
  compose.loadings(pca.1, 5, 'not scaled'),
  compose.loadings(pca.2, 5, 'scaled')
)


ggplot(loadings, mapping=aes(x=t, y=loadings, colour=id)) +
  geom_line() + geom_point() +
  facet_grid(order ~ type)
```



Explaination of the Plot:

- The plot summerizes the loadings of the first five principle components with the scaled (blue) and not-scaled (red) rates.
- The `y` axis stands for the loadings of each original rates in the principle components; `x` axis represents the orignal rates, which are grouped into 4 categories by the labels `non-financial AA`, `non-financial A2P2`, `financial AA` and `asset-backed AA`. Within each group there are 6 maturities: 1, 7, 15, 30, 60, and 90.

Comments:

- The pca with or without scaling the variables can reduce the dimensionalities of the rate change vectors to some extend. For both situations, the first 15 principle components

explain more than 90% of the variance.
- However, they are not as effective as the example we have done in class. I think an important reason is that there are 4 group of rates in this dataset: `non-financial AA`, `non-financial A2P2`, `financial AA` and `asset-backed AA`, as opposed to the in-class example where there is only one rate. The 24 dimensions in this example is actually on a 4-by-6 grid: group, maturity; while in the example we have seen in class, all the 11 dimensions represent different maturites. As a result, it might be harder to compress dimensions for this example.
- PCA using rates without scaling has higher cumulative variance explained than PCA using scaled rates.
- From the loadings plot of first five principle components, we can observe that the 1st and 2nd PCs are approximately the same whether the rates are scaled or not. From the shape of loadings curve we decuce that 1st-PC stands for **parallel shift**, and the 2nd-PC is likely to be **butterfly**.
- PCs after the 2nd do not have clear pattern. And the loadings of `scaled` is very different from that of `not-scaled`.

Part 2:

As described in lecture, one approach to the time series dimension reduction situation we were facing would be to first smooth the time series, and then use these smoothed time series as the input to a dimension reduction algorithm. We will try this here.

In particular, smooth each time series using `loess()`. I will leave the choice of the smoothing parameter up to you. After smoothing, you should use the `predict()` function to evaluate the fitted model on a regular grid of x values. These smoothed time series are what should be utilized in the dimension reduction. Use Isomap. Explore the first two-dimensions to see if there is meaningful low-dimensional structure in the plot.

Comment: If you watched lecture, this will make a lot more sense. Be sure to watch the lecture prior to attempting this.

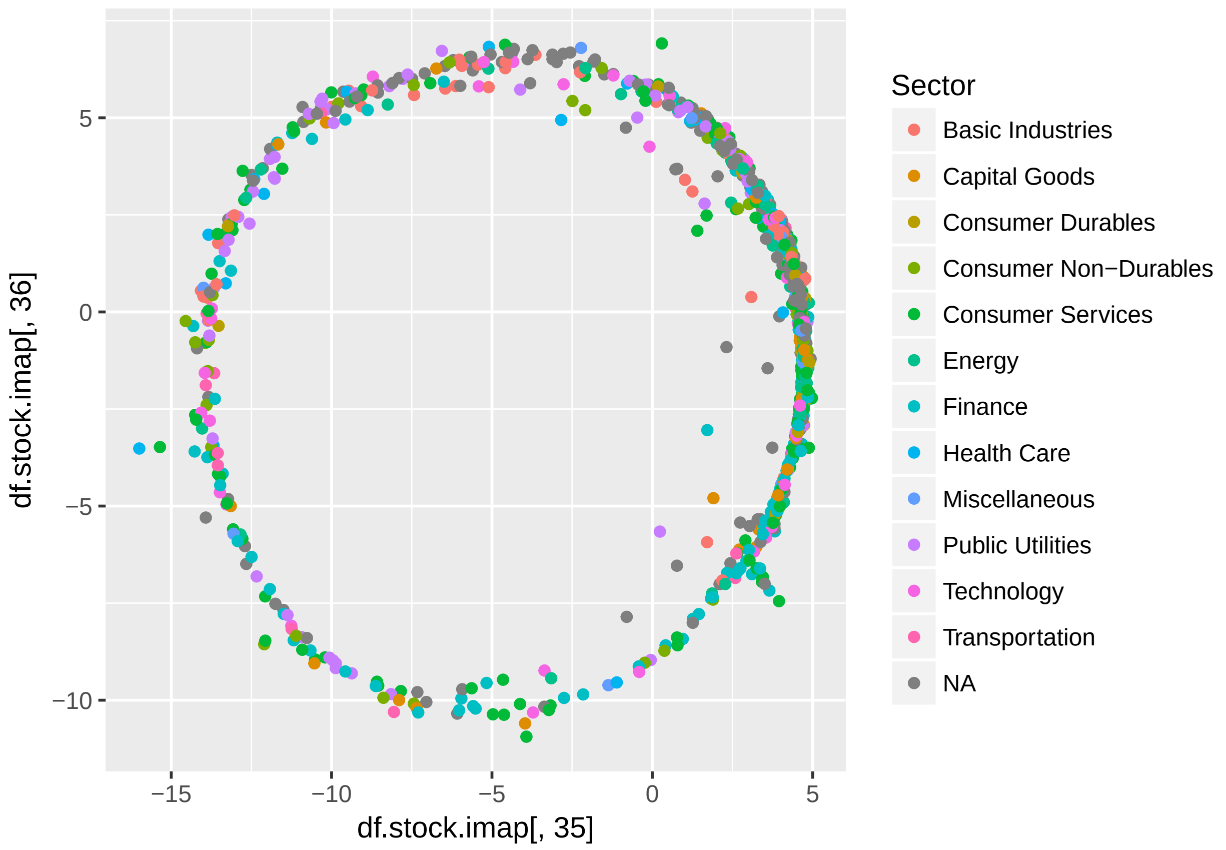
```r
# load the stock data
df.stock = read.table("stocksample.txt", header=T,
                      sep='\t', comment.char="")
df.stock.scale = apply(df.stock[,5:34], 1, scale)
df.stock.scale = as.data.frame(df.stock.scale)
df.stock.scale$time = c(1:30)
```

```r
# function used to construct smoothed stock series
smooth.stock = function(data, degree, span, grid=c(1:30), n=1000) {
  fitted = data.frame(time=grid)
  for (i in c(1:n)) {
    loc.lm = loess(
      df.stock.scale[,i] ~ df.stock.scale$time,
      span=span, degree=degree)
    fitted[,i+1] = scale(predict(loc.lm, grid))
  }
  return(fitted)
}
```

```r
# wrap up function
isomap.wrapper = function(params) {
  df.sm = smooth.stock(
    df.stock.scale,
    degree=params$degree, span=params$span)
  stock.dist = dist(t(df.sm[,2:ncol(df.sm)]))
  fmap = isomap(stock.dist, k=params$k)
  df.stock.imap = cbind(df.stock, fmap$points[,1:params$k])
  plot = ggplot(df.stock.imap,
                aes(x=df.stock.imap[,35],
                    y=df.stock.imap[,36],color=sector)) +
    geom_point() + labs(color='Sector')
  return(list(plot=plot, fmap=fmap, sm=df.sm))
}
```

```r
params.1 = list(
  degree=1,
  span=0.75,
  k=5
)

fmap.1 = isomap.wrapper(params.1)
fmap.1$plot
```



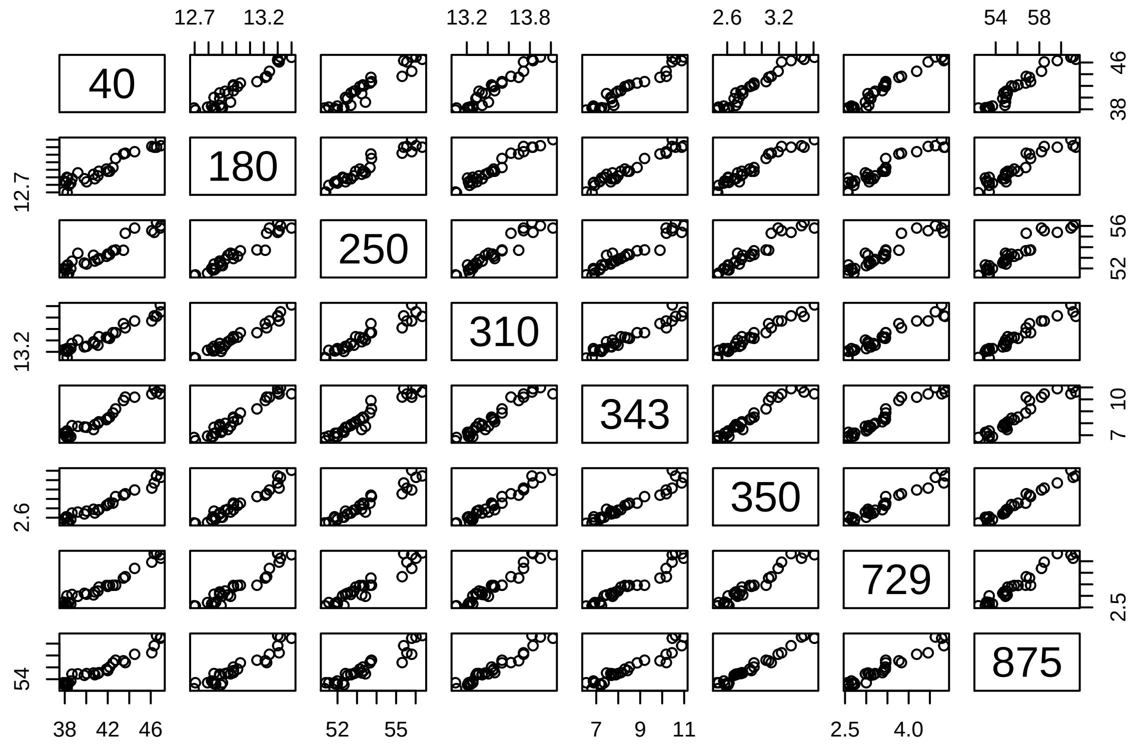
Now we use the `identify` function to check whether a close distance in the isomap representation results in similar pattern in stock time series. I decided to adopt a different approach from what we did in class.

Approach:

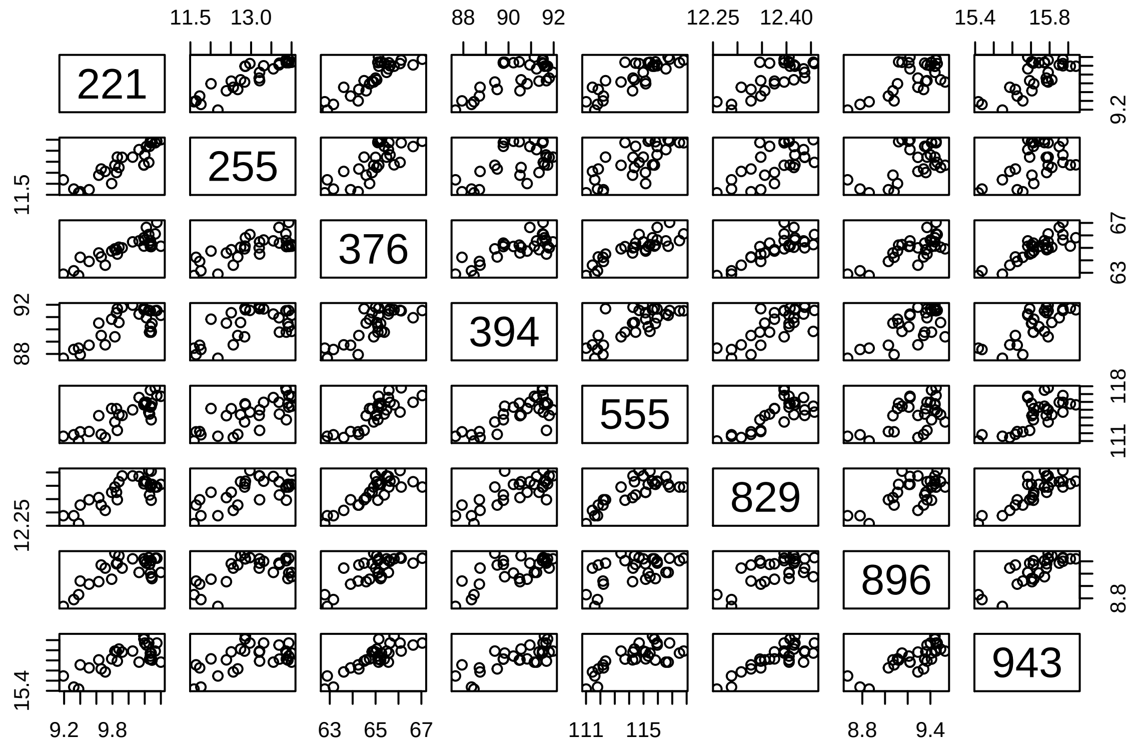
- Choose 2 different regions in the isomap plot above. These two regions are far from each other.
- Treat each region as a "clustering group", within each group, pick several sample observations, here we pick 8 stocks in each group.
- Investigate the correlation matrix of the stock time series of the selected sample. If the first 2 dimensions of `isomap` gives a useful representation of stock time series, then we would observe **similarities within each group**, and **discrepancies across different groups**.

We now do the analysis and investigate the correlation matrix.

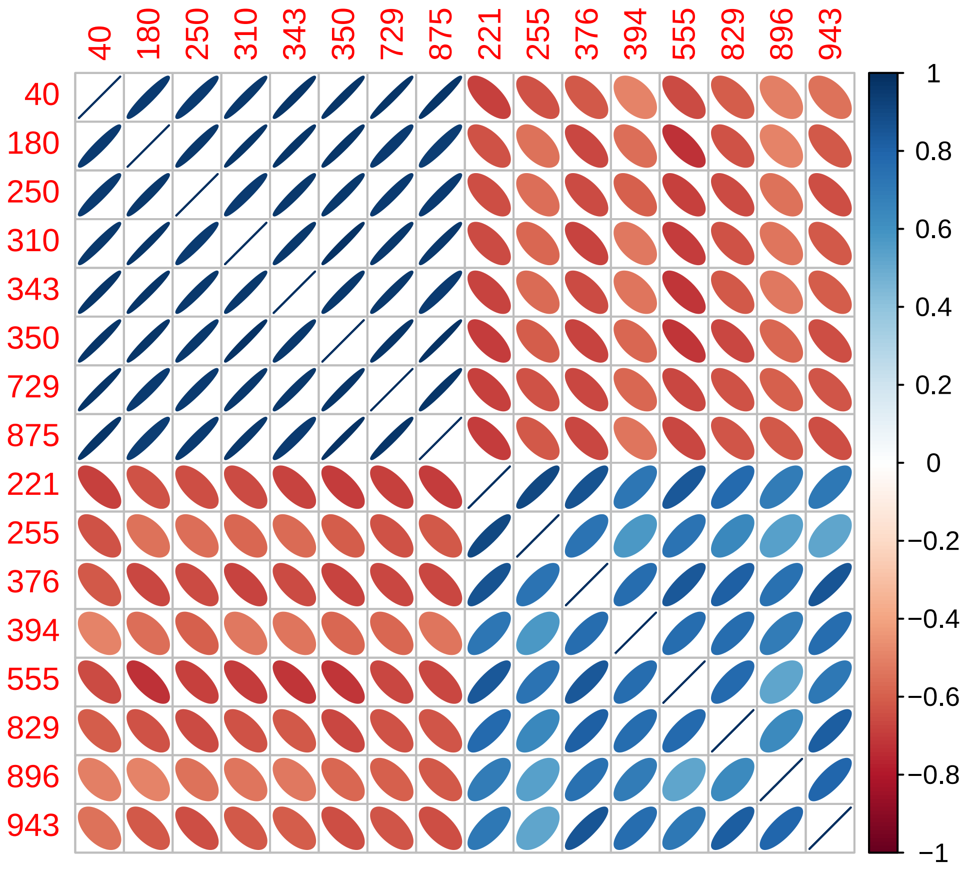
```
df.sm = fmap.1$sm
# plot(fmap.1$fmap$points)
# identify(fmap.1$fmap$points)
g1 = c(40, 180, 250, 310, 343, 350, 729, 875)
g2 = c(221, 255, 376, 394, 555, 829, 896, 943)
plot(as.data.frame(t(df.stock[g1,5:34])))
```



```
plot(as.data.frame(t(df.stock[g2,5:34])))
```



```
df.tmp = as.data.frame(t(df.stock[c(g1, g2),5:34]))
corrplot(cor(df.tmp), method = "ellipse")
```

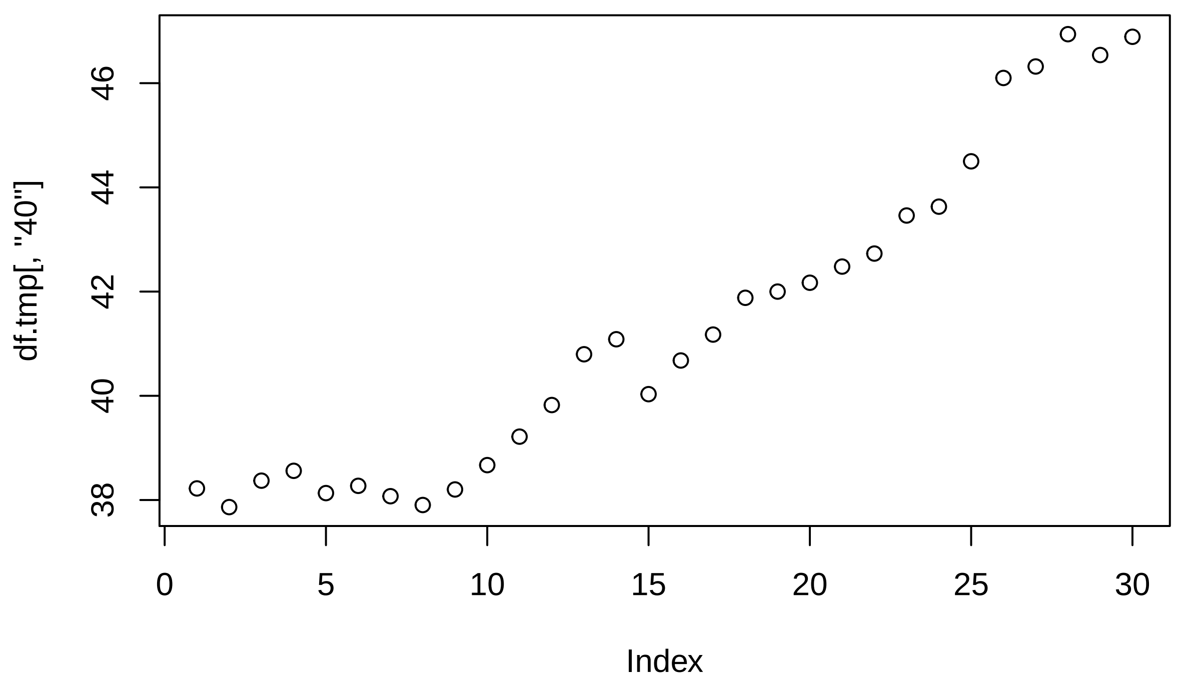


The `corrplot` above is a very straightforward visualization of the correlation matrix. We observe the desired result indeed: within each group, the correlation between stock time series are quite high. Across different groups, there is no such correlation.

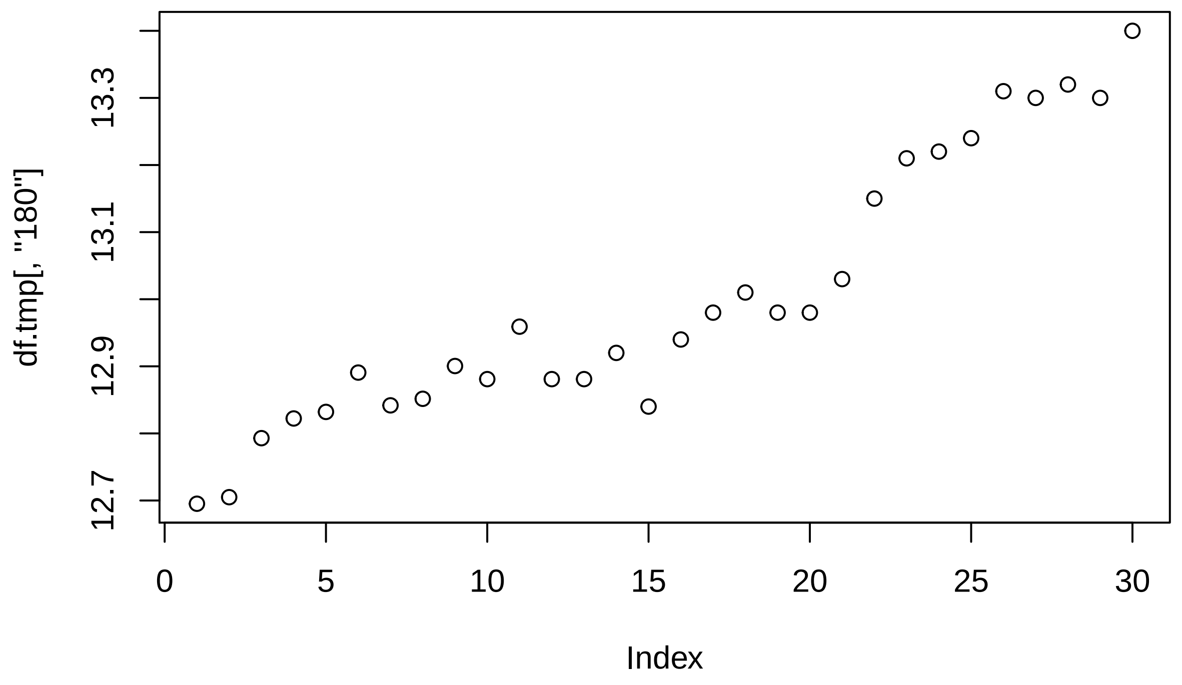
Hence, we conclude that there is a meaningful low-dimensional structure captured by `isomap` indeed.

Further more, we pick 3 points from the first group and plot the time series of each of them:

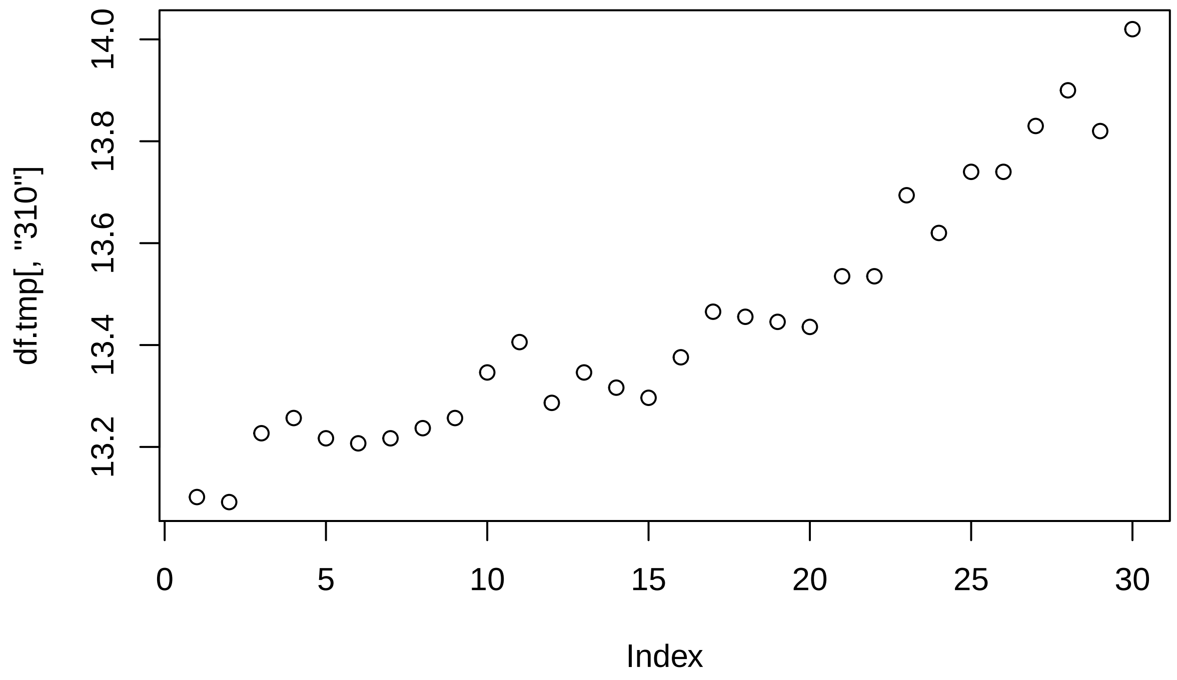
```
plot(df.tmp[,'40'])
```



```r
plot(df.tmp[,'180'])
```



```r
plot(df.tmp[,'310'])
```

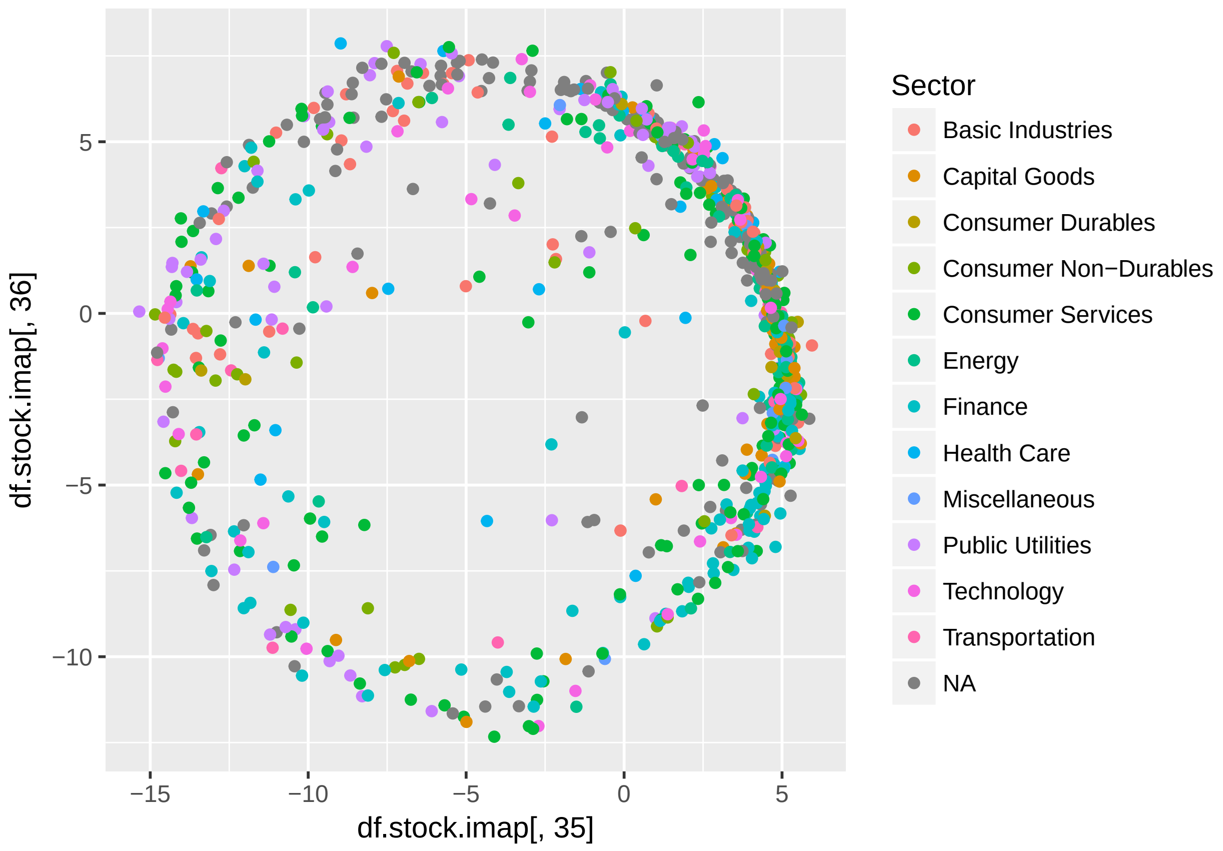


They are very similar, which supports our conclusion.

Lastly, we try different parameters to see whether the shape of the plot of first dimensions change dramatically under different choice of parameters. The answer is no: we obtain similar shapes.

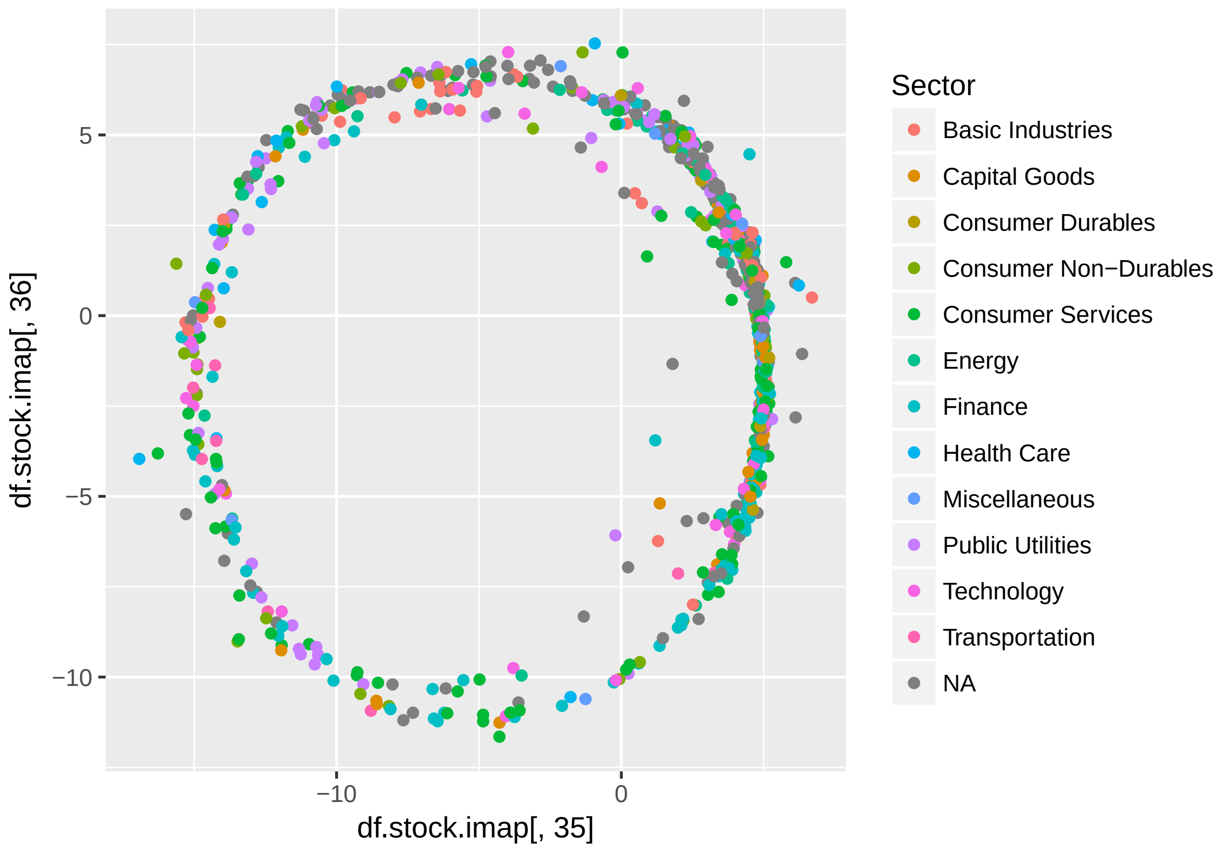
```
params.2 = list(
  degree=2,
  span=0.75,
  k=5
)

fmap.2 = isomap.wrapper(params.2)
fmap.2$plot
```



```
params.3 = list(
  degree=1,
  span=0.75,
  k=4
)

fmap.3 = isomap.wrapper(params.3)
fmap.3$plot
```



df.stock.imap[, 36]
df.stock.imap[, 35]
5
0
−5
−10
−10
0
Sector
Basic Industries
Capital Goods
Consumer Durables
Consumer Non−Durables
Consumer Services
Energy
Finance
Health Care
Miscellaneous
Public Utilities
Technology
Transportation
NA