Stat 897 Spring 2017 Data Analysis Assignment 6

Penn State

Due October 8, 2017

In this assignment we will again use the stock return data (`spreturns.Rda`) from DAA 5 part two. For the purposes of this exercise you may assume the mean returns are zero if it helps.

a) Restrict to the first 100 columns (first 100 stocks) in the data. Compute the covariance matrix of these stocks. Using the `corrplot` package, produce a visual of the matrix. Do not print the covariance matrix. (I will take points off if you do because printing out a 100 x 100 matrix really isn't helpful, visuals are much better!)

```
#C:\\study\\psu\\git\\897\\hw   C:\\study\\897\\hw
setwd("C:\\study\\psu\\git\\897\\hw")
#install.packages('corrplot')
#install.packages('heatmaply')
#install.packages('knitr')

library(heatmaply)
```

```
## Loading required package: plotly
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
## Loading required package: viridis
```

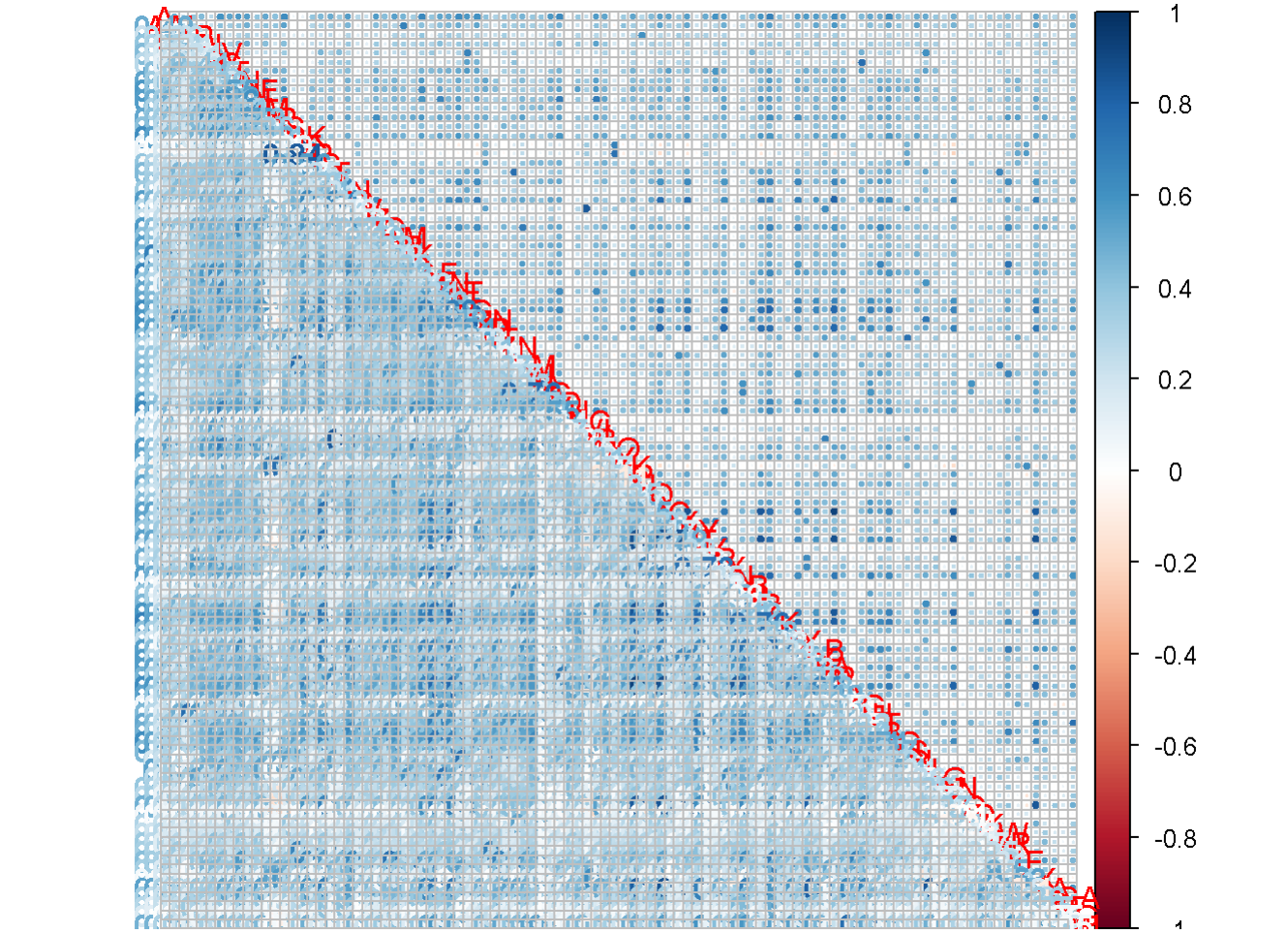
```
## Loading required package: viridisLite
```

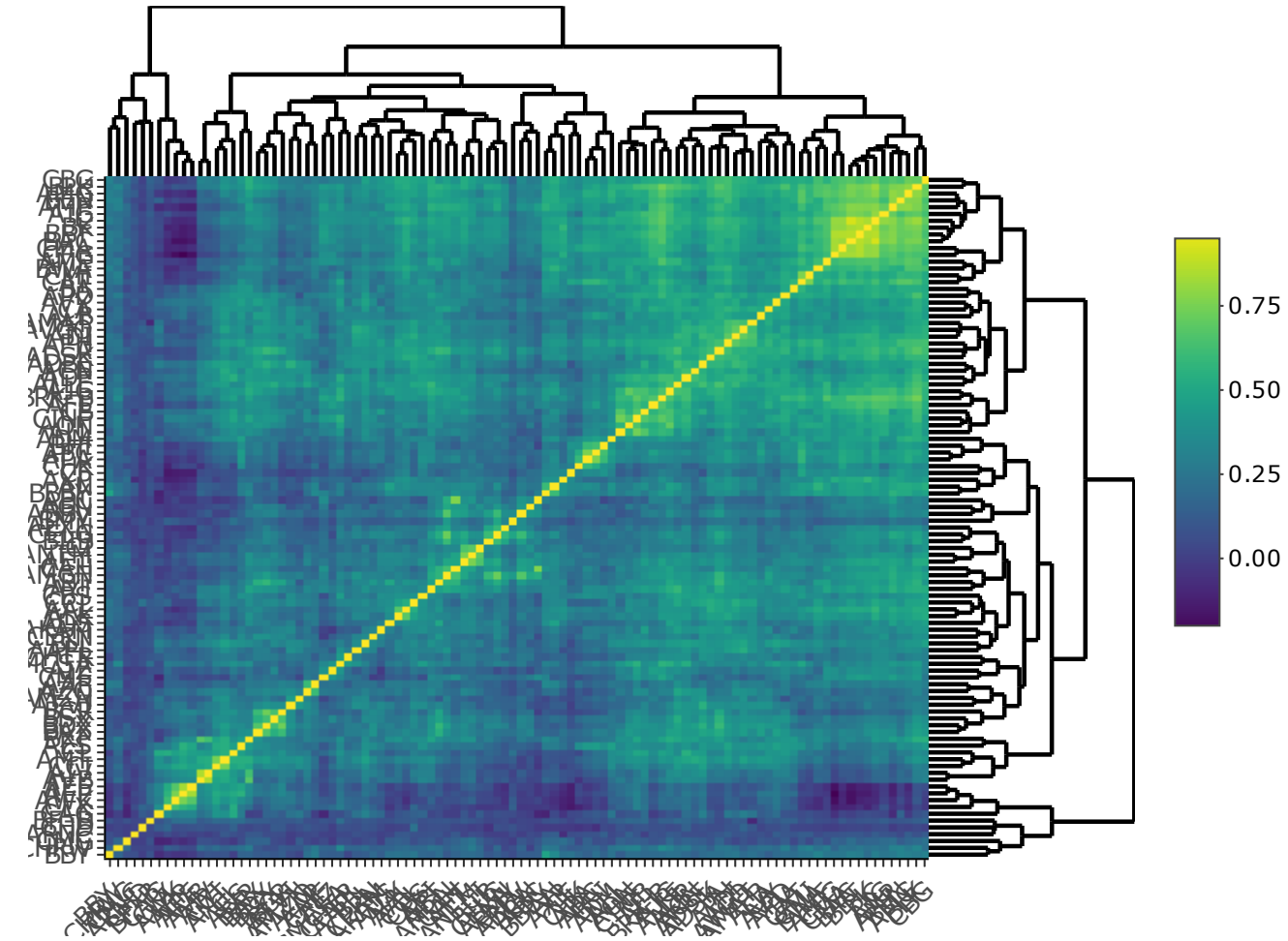
```
##
## ----------------------
## Welcome to heatmaply version 0.11.1
## Type ?heatmaply for the main documentation.
## The github page is: https://github.com/talgalili/heatmaply/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/heatmaply/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(heatmaply))
## ----------------------
```

```
library(corrplot)

load("hw5_spreturns.Rda")
spreturns = spreturns[,1:100]

# Lets scale and center the spreturns object prior to PCA and SVD methods are used.
spreturns_normalized = scale(spreturns, center = TRUE, scale = TRUE)
remove(spreturns)

cov = cov(spreturns_normalized)
R <- cor(spreturns_normalized)
corrplot.mixed(R)
heatmaply(R)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


0.75
0.50
0.25
0.00

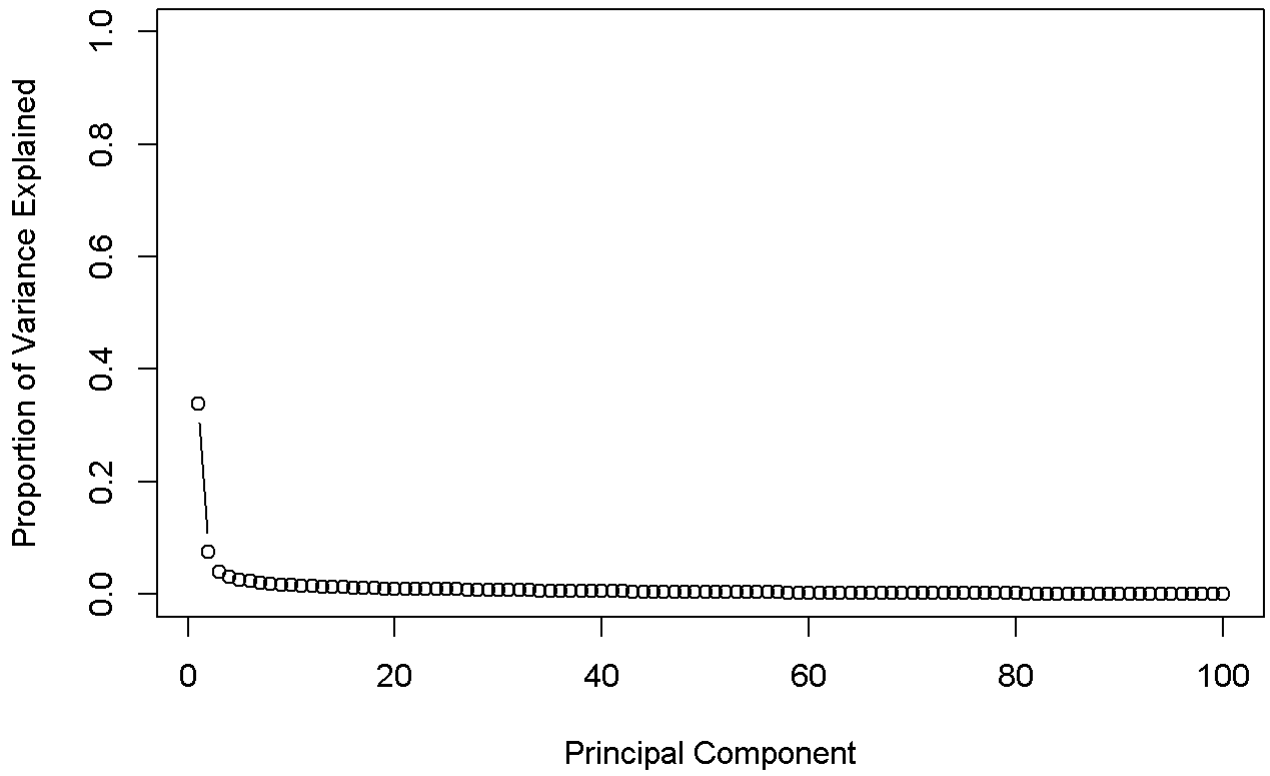
From the cor plot and the heat map we see mostly light blue (in cor plot) and some areas of yellow (in the heatmap). These indicate that the cor/covariance is at high level only for a few elements.

b) Perform PCA on the stocks, and plot the variance explained. Do you see any natural “knee”?

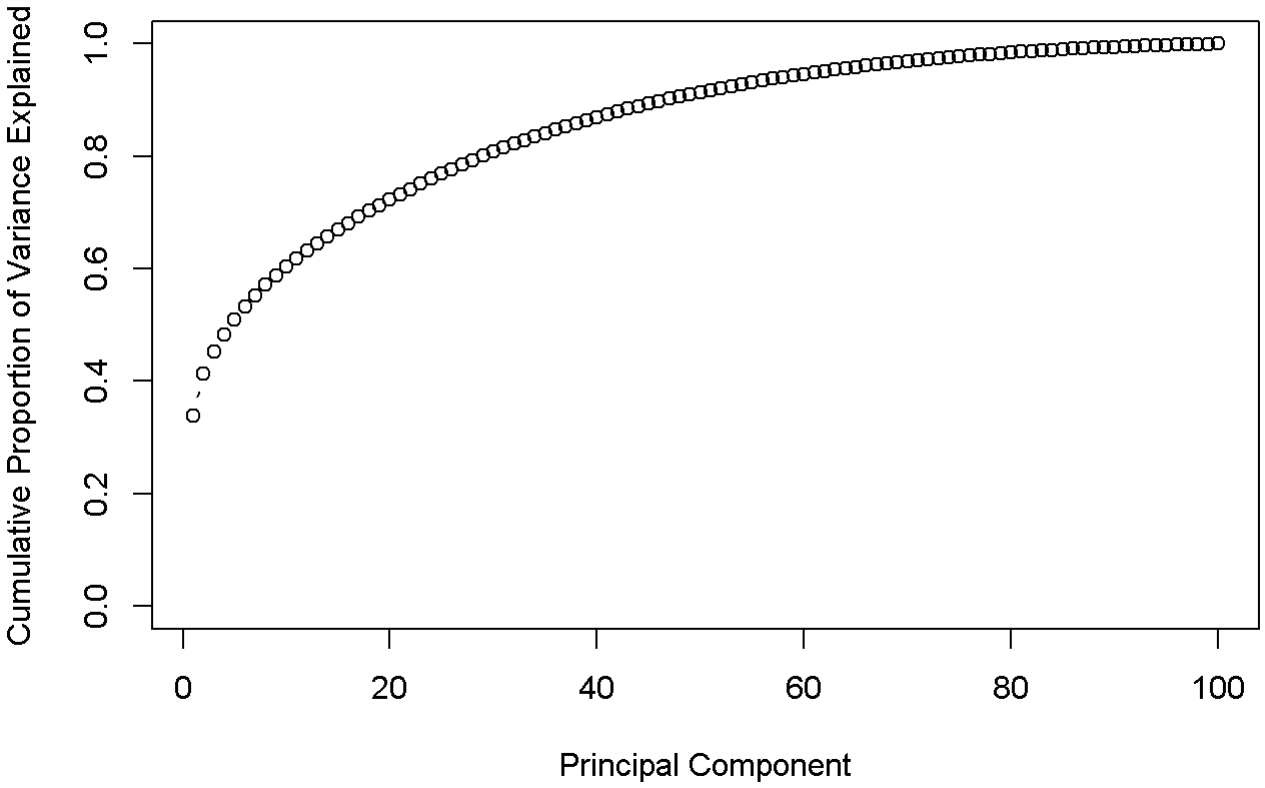
```
pr.out =prcomp (spreturns_normalized)
pr.var =pr.out$sdev ^2
pve=pr.var/sum(pr.var)
pve
```

```
##   [1] 0.3390063400 0.0747286581 0.0389178476 0.0307795269 0.0252345486
##   [6] 0.0240820442 0.0200744857 0.0182234698 0.0162490047 0.0160928553
##  [11] 0.0143067683 0.0137732302 0.0133184874 0.0125499905 0.0120374539
##  [16] 0.0115382908 0.0112407496 0.0106342896 0.0100634027 0.0100008686
##  [21] 0.0096007374 0.0092367685 0.0091836210 0.0090834444 0.0085628749
##  [26] 0.0083857040 0.0083174698 0.0078607343 0.0074783769 0.0072349837
##  [31] 0.0071759335 0.0068771554 0.0066393422 0.0064632480 0.0063273129
##  [36] 0.0062332356 0.0058427929 0.0057370720 0.0054133924 0.0052933354
##  [41] 0.0049815271 0.0049003224 0.0046859222 0.0045786046 0.0045267866
##  [46] 0.0043024937 0.0042286081 0.0041099601 0.0038987423 0.0038015298
##  [51] 0.0036817847 0.0036260999 0.0035995548 0.0033886241 0.0032686164
##  [56] 0.0032210677 0.0031215598 0.0029893774 0.0028102103 0.0027234226
##  [61] 0.0026355237 0.0025304275 0.0024934039 0.0023749587 0.0022937425
##  [66] 0.0021869729 0.0021626950 0.0020407554 0.0020016705 0.0018739961
##  [71] 0.0018501602 0.0018241994 0.0016499450 0.0016116825 0.0015978164
##  [76] 0.0015030446 0.0014937114 0.0014244761 0.0013524421 0.0012839153
##  [81] 0.0012084909 0.0011593995 0.0011306030 0.0011065791 0.0010338775
##  [86] 0.0010188834 0.0009257235 0.0008699957 0.0008082833 0.0007708900
##  [91] 0.0007215452 0.0007007473 0.0006643959 0.0006388312 0.0006291403
##  [96] 0.0005166729 0.0004844461 0.0004731471 0.0004260021 0.0002821447
```

```
plot(pve , xlab="Principal Component", ylab="Proportion of Variance Explained", ylim=c(0,1)
,type='b')
```




```
plot(cumsum (pve ), xlab="Principal Component", ylab ="Cumulative Proportion of Variance Explaine
d",
     ylim=c(0,1), type='b')
```



We do see an elbow / knee in the plot. The first principal component accounts for most of the overall variance (33.9%). The second principal component accounts for far less variance (7.5%) but can be used as compared to the remaining principal component that account for very less marginal variance. First two principal components should be used for further analysis.

c) How would you interpret the factor or first principal component? How would you describe the covariance matrix that results from keeping only the projection onto the first or first and second components?

The first principal component accounts for most of the overall variance (33.9%). The second principal component accounts for far less variance (7.5%) but can be used as compared to the remaining principal component that account for very less marginal variance.

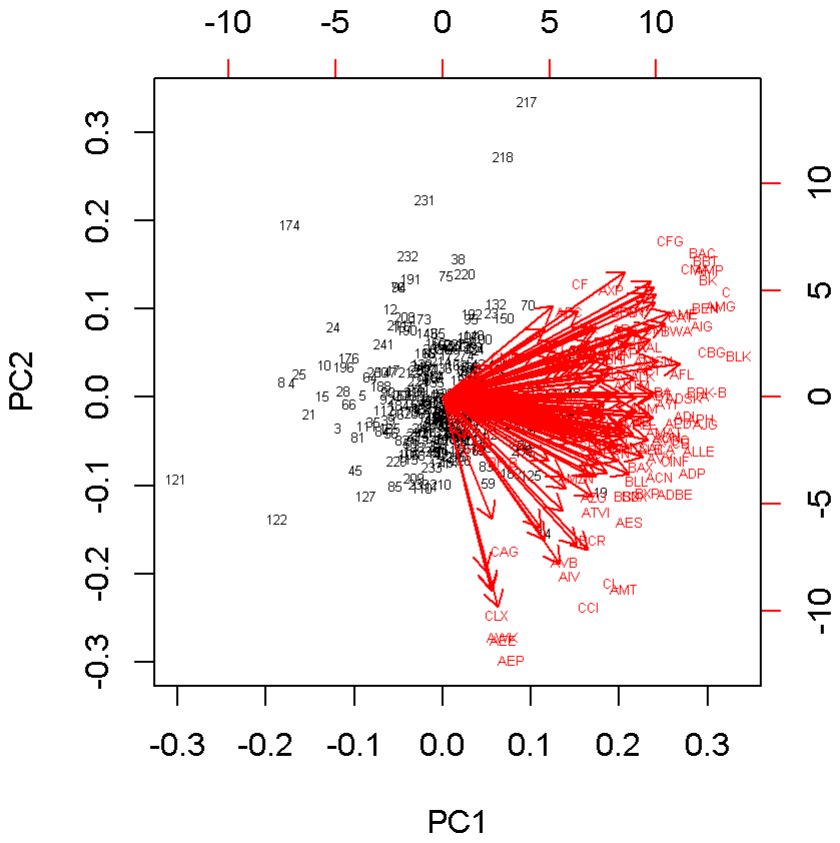
Let's now see the first two components

```
round(pr.out$rotation[,1:2],2)
```

```
##        PC1   PC2
## A     0.13  0.00
## AAL   0.10  0.05
## AAP   0.07 -0.03
## AAPL  0.09 -0.05
## ABBV  0.08  0.05
## ABC   0.06  0.09
## ABT   0.11  0.00
## ACN   0.11 -0.09
## ADBE  0.12 -0.10
## ADI   0.12 -0.02
## ADM   0.10 -0.01
## ADP   0.13 -0.08
## ADS   0.09  0.03
## ADSK  0.12  0.00
## AEE   0.03 -0.26
## AEP   0.03 -0.28
## AES   0.09 -0.13
## AET   0.10  0.03
## AFL   0.12  0.03
## AGN   0.07  0.05
## AIG   0.13  0.08
## AIV   0.06 -0.19
## AIZ   0.08 -0.01
## AJG   0.13 -0.03
## AKAM  0.09 -0.03
## ALB   0.11 -0.06
## ALK   0.10  0.02
## ALL   0.10 -0.03
## ALLE  0.13 -0.06
## ALXN  0.08  0.06
## AMAT  0.11 -0.04
## AME   0.12  0.09
## AMG   0.14  0.10
## AMGN  0.11  0.04
## AMP   0.14  0.14
## AMT   0.09 -0.21
## AMZN  0.07 -0.09
## AN    0.10  0.09
## ANTM  0.10  0.01
## AON   0.11 -0.04
## APA   0.09  0.07
## APC   0.10  0.05
## APD   0.12 -0.03
## APH   0.13 -0.02
## ARNC  0.02  0.00
## ATVI  0.08 -0.12
## AVB   0.06 -0.18
## AVGO  0.12 -0.05
## AVY   0.11 -0.07
## AWK   0.03 -0.26
## AXP   0.09  0.12
## AYI   0.11 -0.01
```

```
## AZO   0.08 -0.11
## BA    0.11  0.01
## BAC   0.13  0.16
## BAX   0.10 -0.07
## BBBY  0.09  0.09
## BBT   0.13  0.15
## BBY   0.06  0.06
## BCR   0.08 -0.15
## BDX   0.10 -0.11
## BEN   0.13  0.10
## BF-B  0.03 -0.07
## BHI   0.09  0.04
## BIIB  0.09  0.05
## BK    0.13  0.13
## BLK   0.15  0.04
## BLL   0.10 -0.09
## BMY   0.05  0.00
## BRK-B 0.13  0.01
## BSX   0.09 -0.11
## BWA   0.12  0.07
## BXP   0.10 -0.10
## C     0.14  0.11
## CA    0.11 -0.05
## CAG   0.03 -0.16
## CAH   0.10  0.06
## CAT   0.12  0.09
## CB    0.12 -0.05
## CBG   0.14  0.05
## CBS   0.10 -0.03
## CCI   0.07 -0.23
## CCL   0.10 -0.02
## CELG  0.09  0.06
## CERN  0.09 -0.06
## CF    0.07  0.12
## CFG   0.12  0.17
## CHD   0.01 -0.06
## CHK   0.07  0.07
## CHRW  0.06 -0.01
## CHTR  0.08 -0.05
## CI    0.08  0.04
## CINF  0.12 -0.07
## CL    0.09 -0.20
## CLX   0.03 -0.23
## CMA   0.13  0.14
## CMCSA 0.10 -0.05
## CME   0.07  0.04
## CMG   0.03  0.04
## CMI   0.11  0.07
```

```
biplot(pr.out, cex=0.4)
```



We find here that the loadings of the first are positive and have similar size, therefore Z1 is almost an average of the variables.

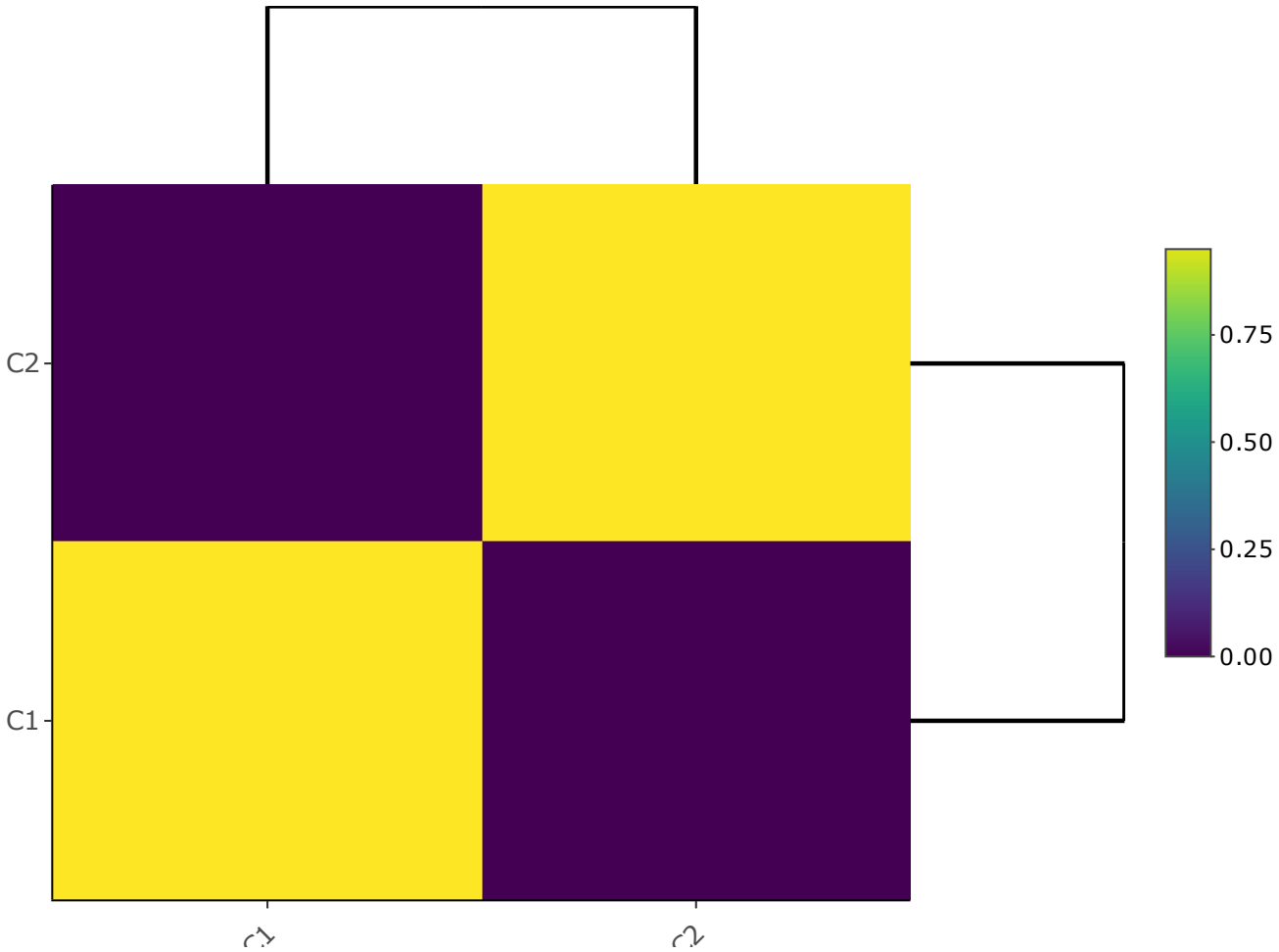
```
nComp = 2
Xhat = pr.out$x[,1:nComp] #%*% t(pr.out$rotation[,1:nComp])
round(cov(Xhat))
```

```
##     PC1 PC2
## PC1  34   0
## PC2   0   7
```

```
R <- cor(Xhat)
corrplot.mixed(R)
heatmaply(R)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```


PC1
PC2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


C2
C1
0.75
0.50
0.25
0.00

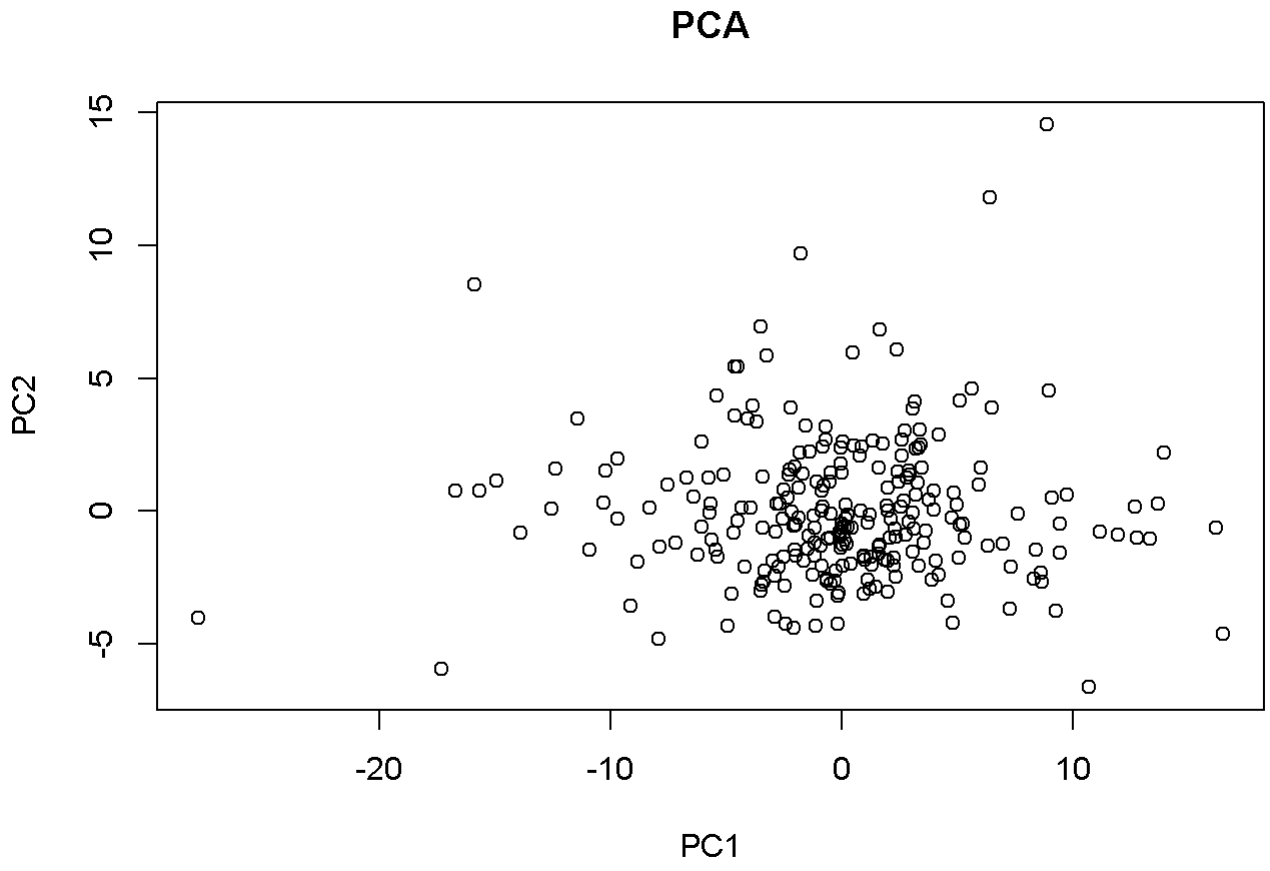
The corr plot is interesting and validates the orthogonality of the principal components. The second principal component, i.e. the second eigenvector, is the direction orthogonal to the first component with the most variance. Because it is orthogonal to the first eigenvector, their projections will be uncorrelated. . In fact, projections on to all the principal components are uncorrelated with each other

We also see that covariance is a diagonal matrix.

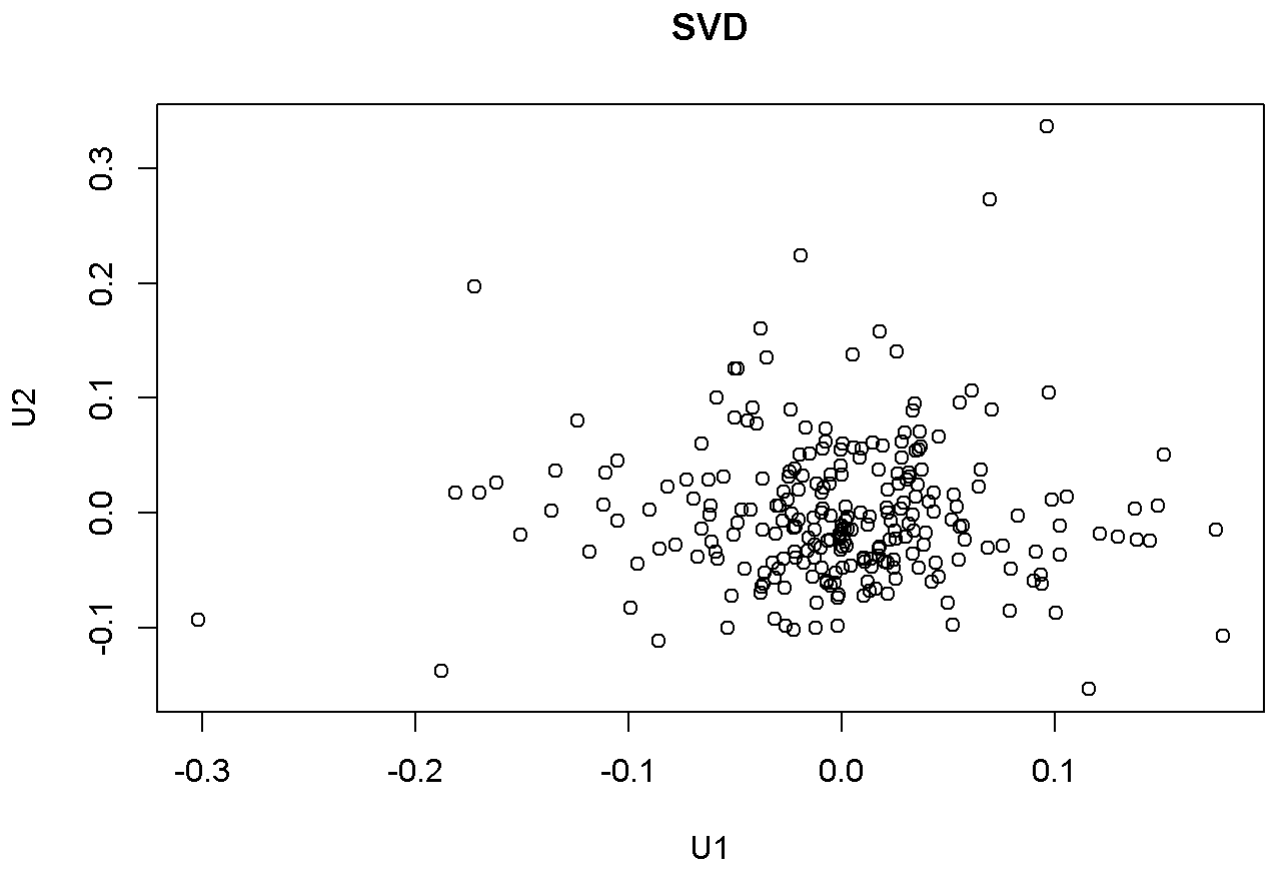
d) Relate the covariance matrix method for PCA (eigendecomposition) to the direct SVD on the given 252 by 100 data matrix X.

Singular values are related to the eigenvalues of covariance matrix. Eigenvalues show the variances of the respective Principal Components.

```
plot(pr.out$x[, 1], pr.out$x[, 2], main = "PCA", xlab = "PC1", ylab = "PC2")
```




```
sv <- svd(spreturns_normalized)
plot(sv$u[, 1], sv$u[, 2], main = "SVD", xlab = "U1", ylab = "U2")
```



PCA is equivalent to performing the SVD on the centered data. Above we see that the plots are exactly the same - Columns of U from the SVD correspond to the principal components x in the PCA.

```
sv$v[1:5, 1:5]
```

```
##            [,1]        [,2]        [,3]        [,4]        [,5]
## [1,] 0.13305699  0.001715931 -0.09549578  0.101398641 -0.03013782
## [2,] 0.10491722  0.054868199  0.01893320  0.143151674 -0.02846162
## [3,] 0.07367211 -0.029891248  0.01696975  0.008464958 -0.07183748
## [4,] 0.08701262 -0.049068542 -0.03432804  0.087941525 -0.04097403
## [5,] 0.07862845  0.046548748 -0.26960912 -0.021314768  0.12884036
```

```
pr.out$rotation[1:5, 1:5]
```

```
##             PC1          PC2         PC3          PC4         PC5
## A    0.13305699  0.001715931 -0.09549578  0.101398641 -0.03013782
## AAL  0.10491722  0.054868199  0.01893320  0.143151674 -0.02846162
## AAP  0.07367211 -0.029891248  0.01696975  0.008464958 -0.07183748
## AAPL 0.08701262 -0.049068542 -0.03432804  0.087941525 -0.04097403
## ABBV 0.07862845  0.046548748 -0.26960912 -0.021314768  0.12884036
```

```
round(sv$v[1:5, 1:5] - pr.out$rotation[1:5, 1:5])
```

```
##       PC1 PC2 PC3 PC4 PC5
## A       0   0   0   0   0
## AAL     0   0   0   0   0
## AAP     0   0   0   0   0
## AAPL    0   0   0   0   0
## ABBV    0   0   0   0   0
```

If matrix M is our data matrix (mXn) m rows represents our data points and and the n columns represents the features of the data point.

SVD: Since the data matrix M will not have exactly the same number of data points as features (i.e. mXn) the matrix M is not a square matrix and a diagonalization of the from M=USUT where U is an m×m orthogonal matrix of the eigenvectors of M and S is the diagonal m×m matrix of the eigenvalues of M will not exist. However, for such cases where nXm, a diff decomposition is possible and M can be factored as follows M=USVT, where

1. U is an m×m orthogonal matrix of the the “left singular-vectors” of M.
2. V is an n×n orthogonal matrix of the the “right singular-vectors” of M.
3. And, S is an m×n matrix with non-zero diagonal entries referred to as the “singular-values” of M.

PCA: PCA uses the symmetric n×n covariance matrix of non-symmetric M i.e MTM. Because MTM is symmetric it is diagonalizable. So PCA works by finding the eigenvectors of the covariance matrix and ranks them by their respective eigenvalues. The eigenvectors with the greatest eigenvalues are the Principal Components of the data matrix.

Some matrix algebra can be done to show that the Principal Components of a PCA diagonalization of the covariance matrix MTM are the same left-singular vectors that are found through SVD (i.e. the columns of matrix V):

From SVD we have M=USVT so…

. MTM=(USVT)T(USVT) . MTM=(VSTUT)(USVT) . but since U is orthogonal UTU=I so . MTM=VS2VT

where S2 is an n×n diagonal matrix with the diagonal elements S2 from the matrix S. So the matrix of eigenvectors V in PCA are the same as the singular vectors from SVD, and the eigenvalues generated in PCA are just the squares of the singular values from SVD.