Howework2

Yi Chen

1/30/2020

Homework 2

Step 1

Read the data

```
library(dplyr)
library(readxl)
options(digits=2)
skills2020 <- read_excel("HUDM 5124_class skills data_2016-2020.xlsx")
# summary(skills2020)
```

Set the skill matrix, replace the NA with 0, and delete the 5th column. There is another column which is not numerical since there is a value in the column is string. I repalce that string value as 1 and transfer the type of column into numberic.

```
sk <- skills2020 %>%
      replace(., is.na(.), 0) %>%  ## repalce all the missing data as 0
      select(7:(ncol(skills2020)-1)) %>%  ## seiect the skill columns
      mutate(OTHER = replace(OTHER,! OTHER  %in% c('0','1'),'1')) %>%  # replac
e the OTHER columns
      mutate_if(is.character,as.numeric) %>% ## make all columns as numberic
      select_if(function(col) is.numeric(col) && mean(col)>0) %>%
      mutate_if(is.numeric, ~1 * (. > 0))  ##repalce all the number bigger than
1 as 1
#summary(sk)
```

Get the correlation matrix (23 time 23 since I keep the column “OTHER”)

```
Rxc <- cor(sk)
#round(Rxc,2)
```

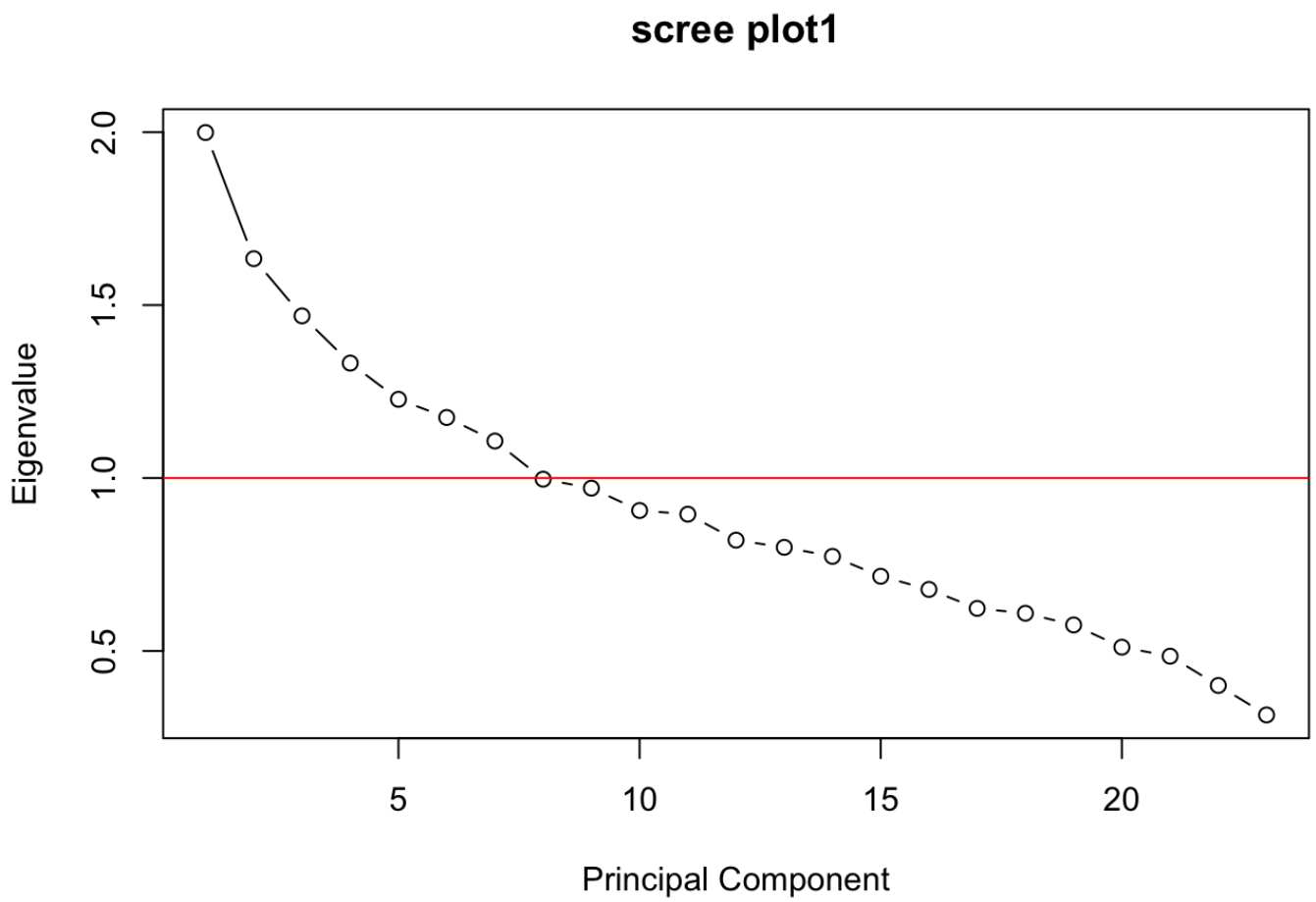
step 2

A. use princomp to run a PCA

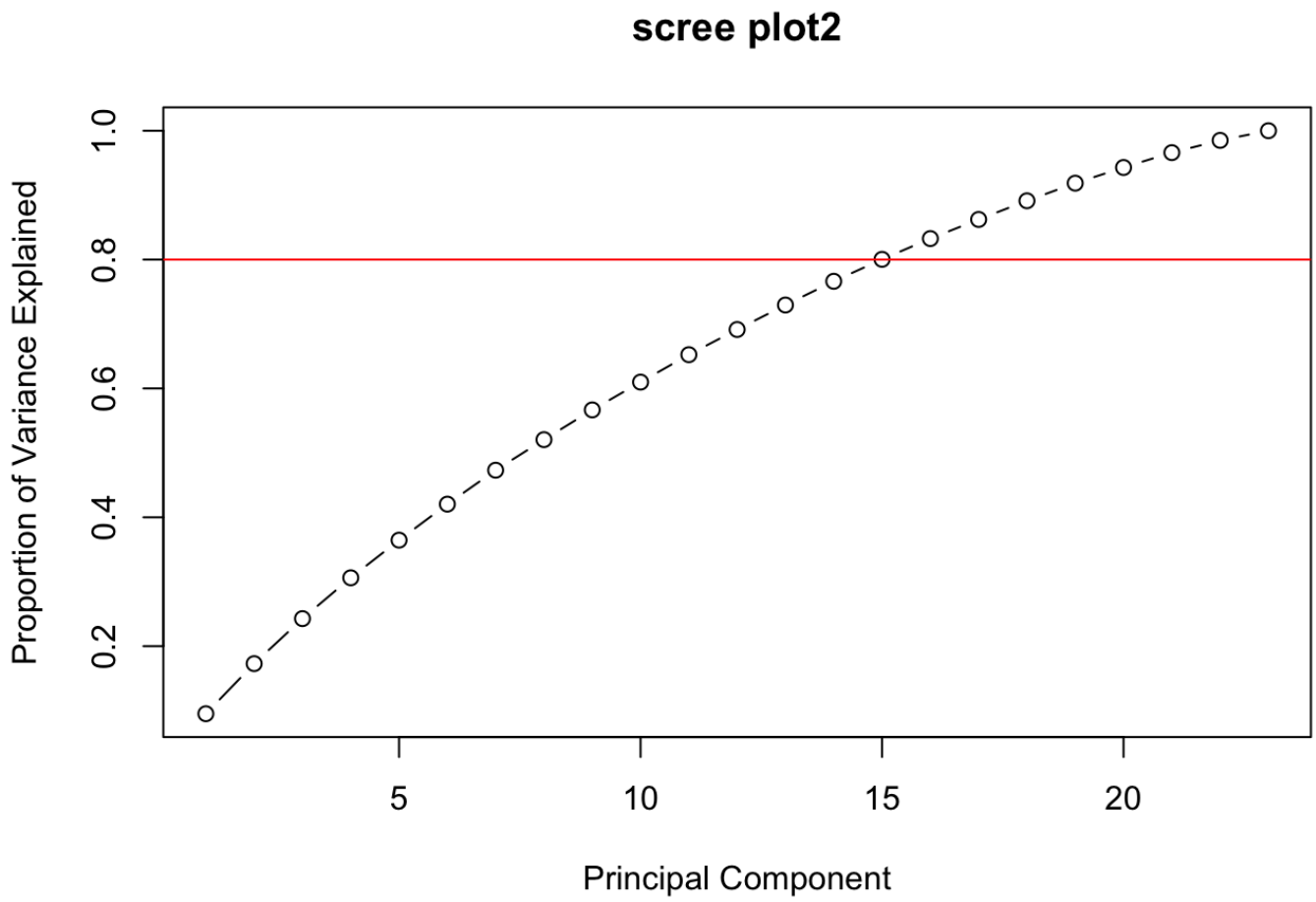
```
princ <- princomp(x = sk,cor = T, scores = T)
print(princ$sdev,digits = 2)
```

```
##  Comp.1  Comp.2  Comp.3  Comp.4  Comp.5  Comp.6  Comp.7  Comp.8  Comp.9 Com
p.10
##    2.00    1.63    1.47    1.33    1.23    1.18    1.11    1.00    0.97
0.91
## Comp.11 Comp.12 Comp.13 Comp.14 Comp.15 Comp.16 Comp.17 Comp.18 Comp.19 Com
p.20
##    0.90    0.82    0.80    0.77    0.72    0.68    0.62    0.61    0.58
0.51
## Comp.21 Comp.22 Comp.23
##    0.48    0.40    0.32
```

```
plot(princ$sdev, xlab = "Principal Component",ylab = "Eigenvalue",type = "b", m
ain = 'scree plot1')
abline(h=1,col='Red')
```



```
plot(cumsum(princ$sdev)/sum(princ$sdev), xlab = "Principal Component",ylab = "P
roportion of Variance Explained",type = "b", main = 'scree plot2')
abline(h=0.8,col='Red')
```



If we only select the principal components which has the standard deviation bigger than 1. Then, the number of principal should be 7. Some people also chooce the number of components which can at least cover 80% of the total variance. In this case, the number of component will be 15.

I personally prefer to chooce a smaller number of principal component since it will realize the popurpose of simplify the model and easier for interpretation. In this case, since almost all principal component covers only a small number of variance, it is hard to really use the PCA to reduce the dimension. I choose to pick 3 principal component, which I recognize that is very risk, since it only capture 24.3% of total variance. I choose 3 mianly because it is easiler to discuss the result and interprete it in the following questions. Another reason is that the decrease of eignvalue from principal 3 to 4 is slower than the decrase from principal 2 to 3. To some extent, this means what the marginal benefit of adding a extra principal tends to be small. Consequently, there is no strong necessary to add more principal.

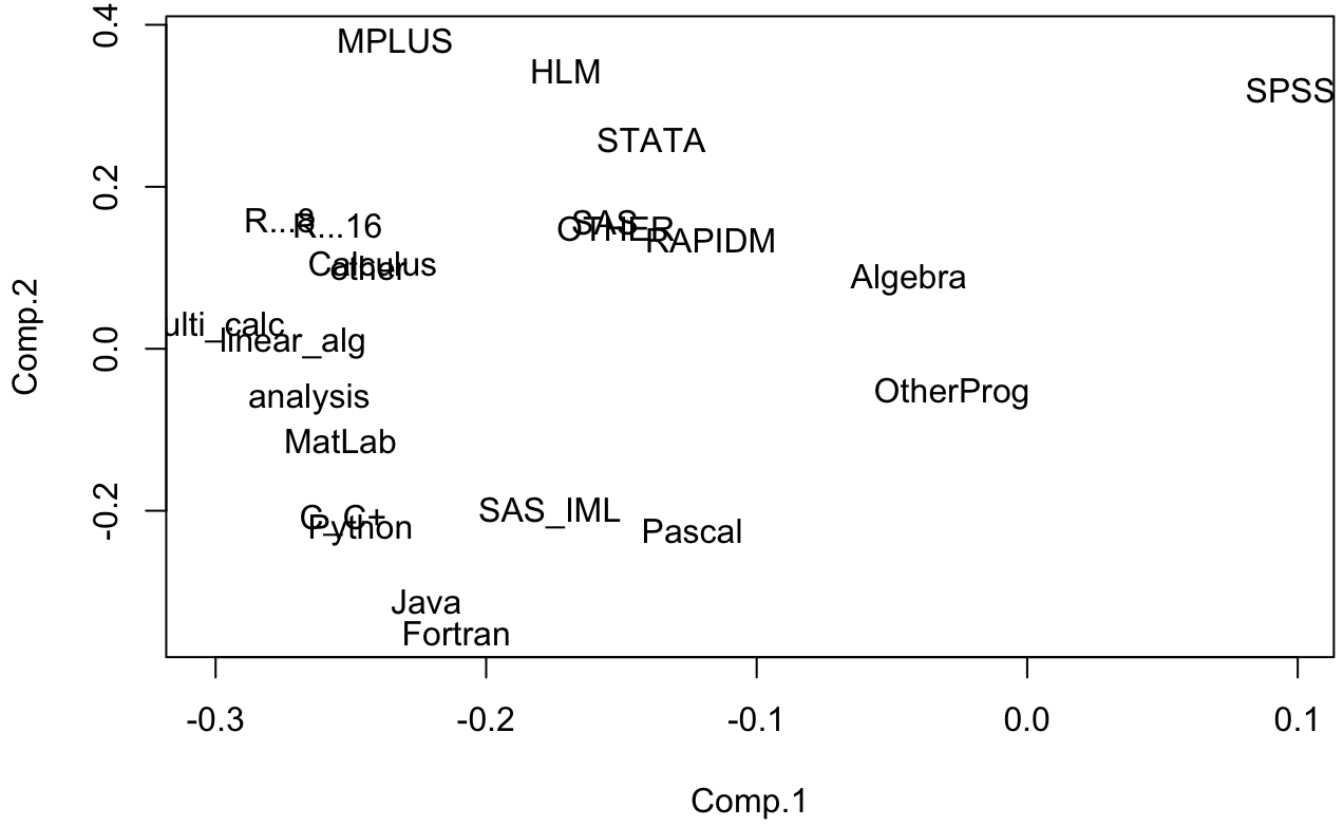
B.

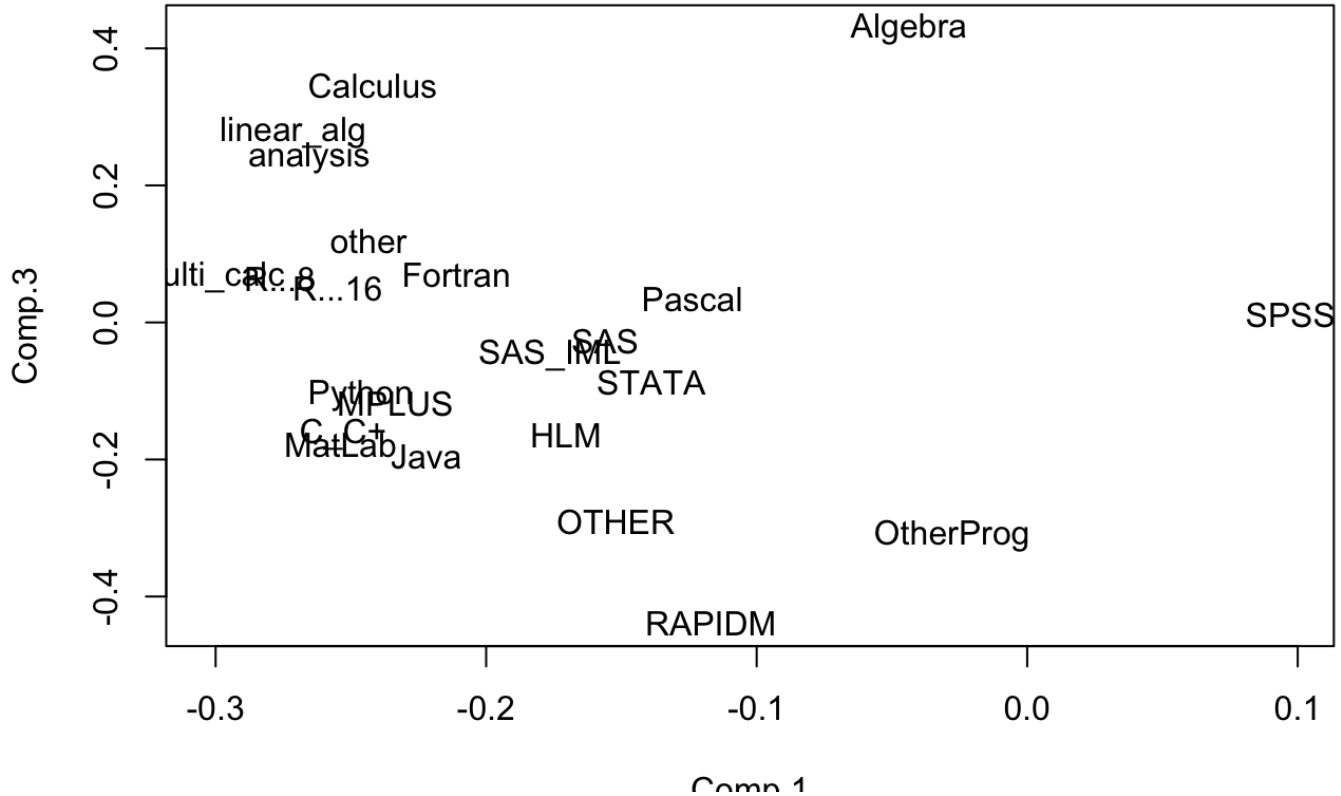
```
## print out the first 3 components
p3 <- princ$loadings[,1:3]
print(p3)
```

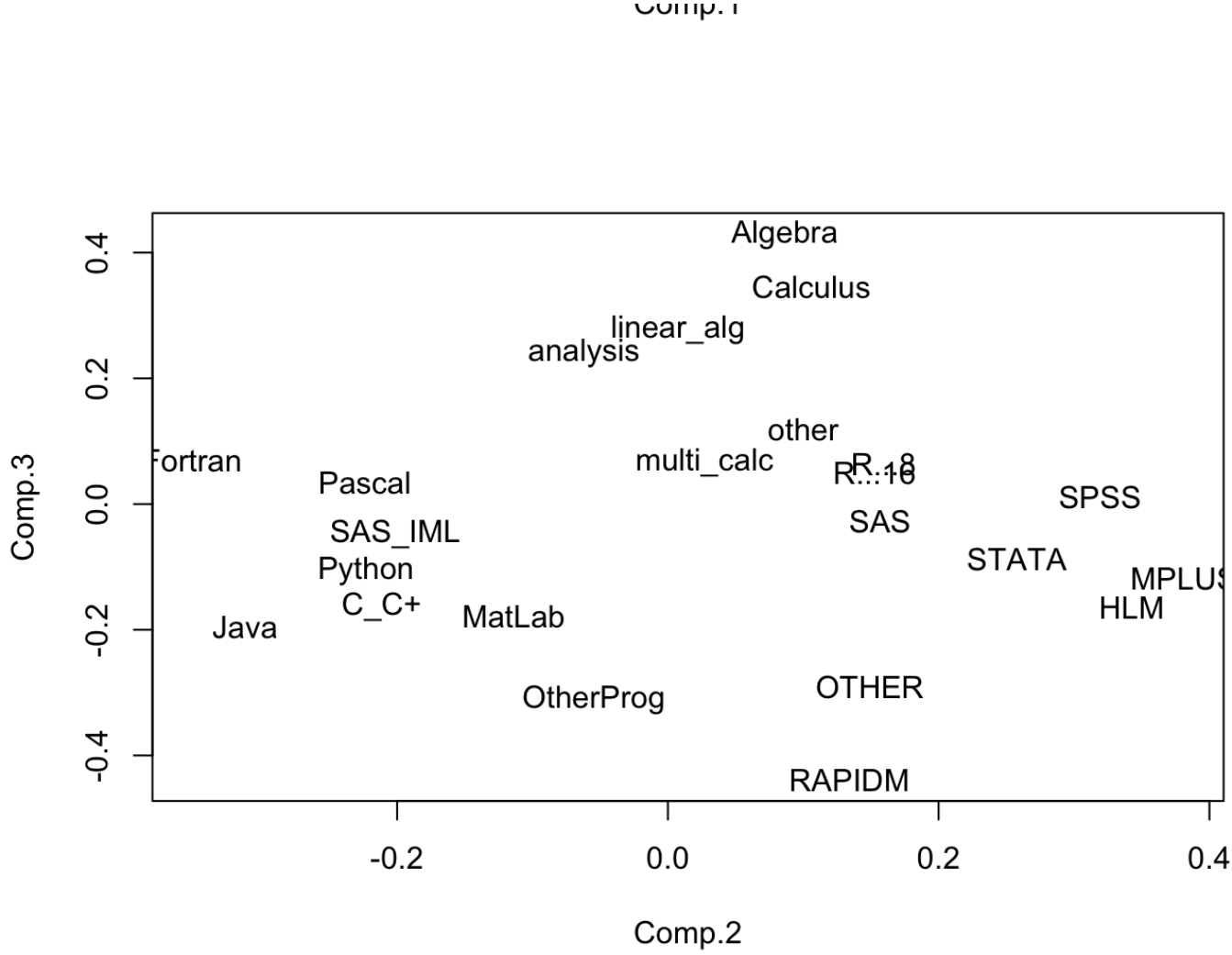
```
##            Comp.1  Comp.2 Comp.3
## SPSS        0.097  0.3192  0.011
## R...8      -0.276  0.1594  0.064
## SAS        -0.156  0.1568 -0.026
## STATA      -0.139  0.2579 -0.087
## RAPIDM     -0.117  0.1340 -0.438
## MPLUS      -0.234  0.3812 -0.118
## HLM        -0.171  0.3427 -0.164
## OTHER      -0.152  0.1492 -0.290
## R...16     -0.255  0.1528  0.050
## SAS_IML    -0.176 -0.2014 -0.047
## MatLab     -0.254 -0.1139 -0.179
## Java       -0.222 -0.3123 -0.195
## C_C+       -0.253 -0.2113 -0.162
## Python     -0.246 -0.2230 -0.106
## Pascal     -0.124 -0.2238  0.034
## Fortran    -0.211 -0.3514  0.071
## OtherProg  -0.028 -0.0547 -0.310
## Algebra    -0.044  0.0863  0.428
## Calculus   -0.242  0.1060  0.346
## multi_calc -0.302  0.0271  0.067
## linear_alg -0.271  0.0074  0.278
## analysis   -0.265 -0.0619  0.240
## other      -0.243  0.1000  0.120
```

Note: since I pick 3 components, there are 6 paris.

```
possible_combination <- combn(3,2)
for (c in 1:ncol(possible_combination)){
  combination <- possible_combination[,c]
  cur_pair <- p3[,combination]
  plot(cur_pair,pch = '')
  text(cur_pair,colnames(sk))
}
```


MPLUS
HLM
SPSS
STATA
R...8
R...16
SAS
OTHER
RAPIDM
Calculus
other
Algebra
ulti_calc
linear_alg
analysis
OtherProg
MatLab
C, C++
SAS_IML
Python
Pascal
Java
Fortran
0.4
0.2
0.0
-0.2
Comp.2
-0.3
-0.2
-0.1
0.0
0.1
Comp.1


Algebra
Calculus
linear_alg
analysis
other
ulti_calc
R...8
R...16
Fortran
Pascal
SPSS
SAS
SAS_IML
STATA
Python
MPLUS
C, C++
HLM
MatLab
Java
OTHER
OtherProg
RAPIDM
0.4
0.2
0.0
-0.2
-0.4
Comp.3
-0.3
-0.2
-0.1
0.0
0.1
Comp.1



Interpretation: Component 1 is SPSS since it is the only positive number.

Component 2 is traditional software skill which has bigger value for the close-source softwares like Mplus, SPSS, and Stata. It also have big positive number for the math skills like Algebra and Calculus.

Component 3 is new software skill which has bigger number for new open-source softwares like python and R.

C.

```
skill.eigen <- eigen(cor(sk))
skill.eigen$values
```

```
##  [1] 3.996 2.671 2.158 1.775 1.507 1.381 1.226 0.993 0.942 0.821 0.802 0.673
## [13] 0.639 0.598 0.513 0.460 0.388 0.371 0.331 0.261 0.235 0.160 0.099
```

```
#round(skill.eigen$vectors,2)
```

Assume that I pick 3 component as what I did in question B

```
# compute the principal component from eignvalue and eigenvector
wgt <-diag(3)
for (i in 1:3){
  wgt[i,i] <- sqrt(skill.eigen$values[i])
}
wgt
```

```
##      [,1] [,2] [,3]
## [1,]    2  0.0  0.0
## [2,]    0  1.6  0.0
## [3,]    0  0.0  1.5
```

```
p3_eigen <- skill.eigen$vectors[,1:3] %*% wgt
p3_eigen
```

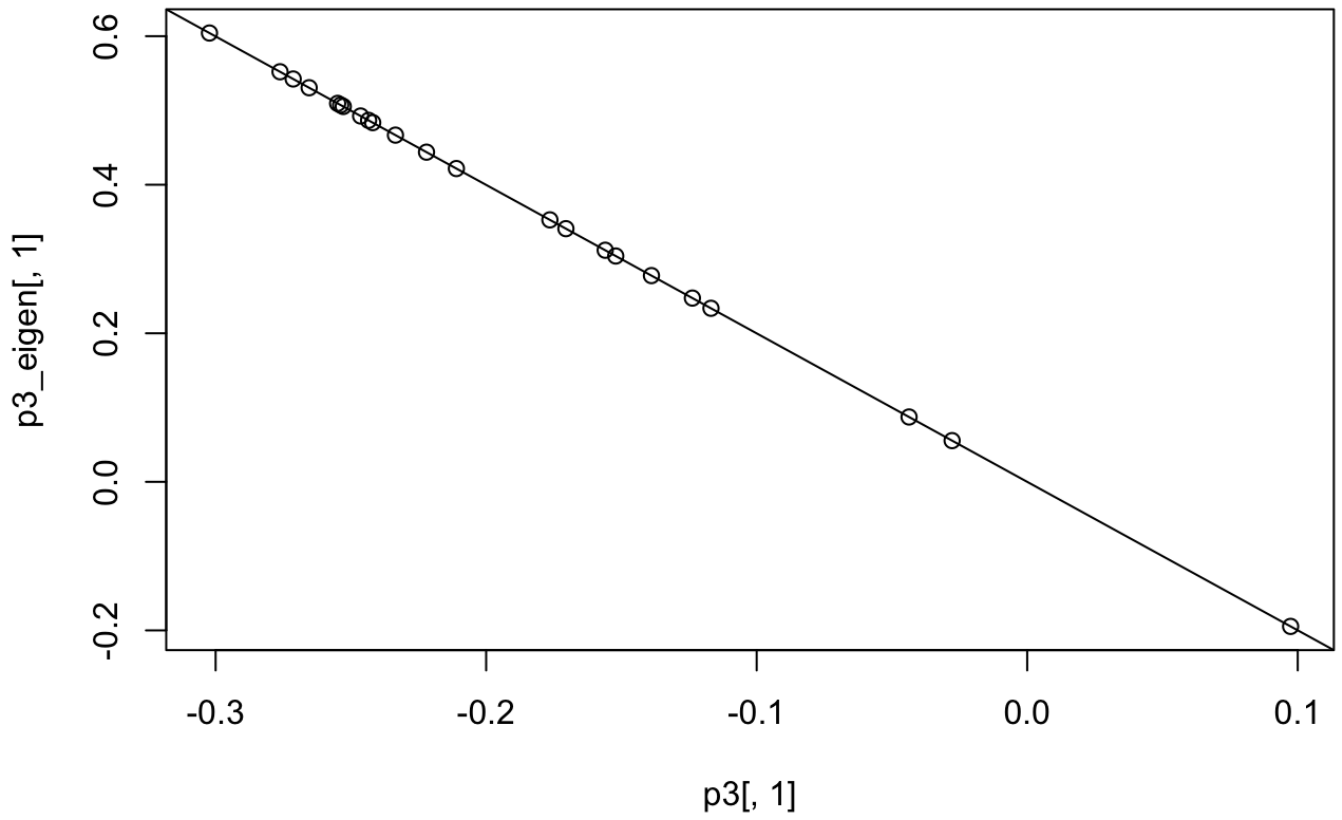
```
##         [,1]   [,2]   [,3]
##  [1,] -0.195  0.522  0.017
##  [2,]  0.552  0.261  0.095
##  [3,]  0.312  0.256 -0.038
##  [4,]  0.278  0.421 -0.127
##  [5,]  0.234  0.219 -0.643
##  [6,]  0.467  0.623 -0.173
##  [7,]  0.341  0.560 -0.240
##  [8,]  0.304  0.244 -0.426
##  [9,]  0.510  0.250  0.074
## [10,]  0.353 -0.329 -0.069
## [11,]  0.507 -0.186 -0.263
## [12,]  0.444 -0.510 -0.286
## [13,]  0.505 -0.345 -0.238
## [14,]  0.493 -0.364 -0.156
## [15,]  0.247 -0.366  0.050
## [16,]  0.422 -0.574  0.104
## [17,]  0.055 -0.089 -0.456
## [18,]  0.087  0.141  0.629
## [19,]  0.484  0.173  0.508
## [20,]  0.604  0.044  0.099
## [21,]  0.542  0.012  0.408
## [22,]  0.531 -0.101  0.353
## [23,]  0.487  0.163  0.176
```

Compare the result from princomp and eigenvalue eigenvector

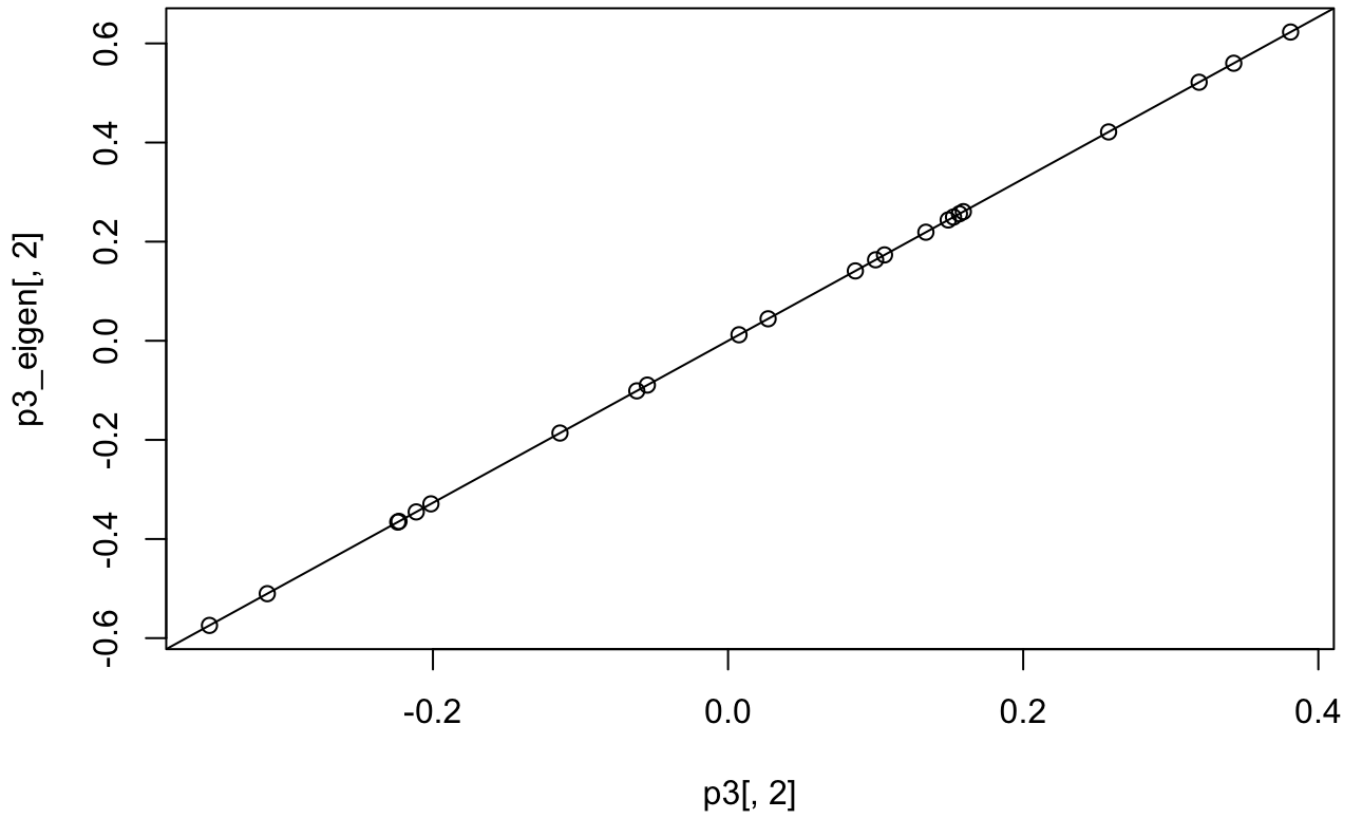
```
# linear corelation
c(cor(p3[,1],p3_eigen[,1]),cor(p3[,2],p3_eigen[,2]),cor(p3[,3],p3_eigen[,3]))
```

```
## [1] -1  1  1
```

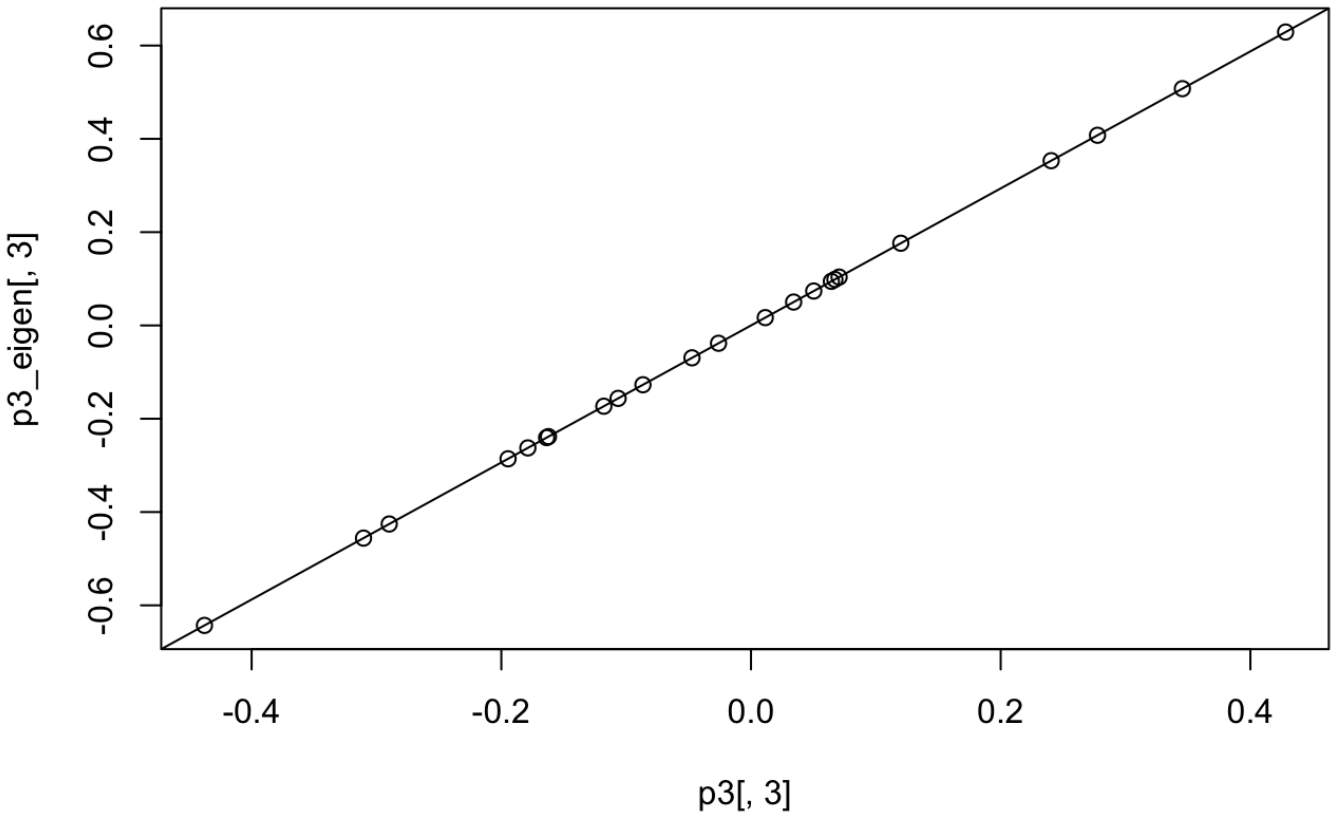
```
plot(p3[,1],p3_eigen[,1])
abline(lm(p3_eigen[,1]~p3[,1]))
```




```
plot(p3[,2],p3_eigen[,2])
abline(lm(p3_eigen[,2]~p3[,2]))
```




```
plot(p3[,3],p3_eigen[,3])
abline(lm(p3_eigen[,3]~p3[,3]))
```



Even thouth the result are not the same from eigenvalue/eigenvecotr and princomp. The linear correlation between them are perfect. Which means the result are only differ in linear transformation which means they are fundamentally the same.

rotation rutine

```
p3vm <- varimax(p3_eigen,normalize = T,eps = 1e-5)
p3vm$loadings
```

```
##
## Loadings:
##       [,1]   [,2]   [,3]
##  [1,]  0.223  0.510
##  [2,]  0.513 -0.158  0.306
##  [3,]  0.393
##  [4,]  0.511
##  [5,]  0.505 -0.119 -0.497
##  [6,]  0.788  0.110
##  [7,]  0.683  0.134
##  [8,]  0.500 -0.106 -0.268
##  [9,]  0.484 -0.142  0.271
## [10,]       -0.485
## [11,]  0.289 -0.521
## [12,]       -0.718 -0.148
## [13,]  0.171 -0.630
## [14,]  0.124 -0.620
## [15,] -0.103 -0.419  0.106
## [16,] -0.147 -0.676  0.201
## [17,]  0.118 -0.189 -0.412
## [18,]        0.165  0.628
## [19,]  0.279         0.659
## [20,]  0.398 -0.349  0.311
## [21,]  0.239 -0.272  0.574
## [22,]  0.170 -0.356  0.510
## [23,]  0.378 -0.170  0.351
##
##                [,1] [,2] [,3]
## SS loadings    3.20 3.23 2.40
## Proportion Var 0.14 0.14 0.10
## Cumulative Var 0.14 0.28 0.38
```