Assignment 5 [R]\_Matrix\_Operations

## INTRODUCTION

**IMPORTANT NOTE**: In this assignment I have used three Data sets, Just out of curiosity. Two data set with Int values, and one with Float values and having NA’s in it. And I leaned a lot from the 3rd data set with NA’s.

Now, for this Assignment we only need two libraries, one from which we have taken the Data sets. And the one to do Matrix operations.

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.4 v dplyr 1.0.7  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 2.0.1 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(matlib)

## DATABASE INFORMATION AND FILTERING

Here, we have three data sets. Attitude with all numeric value. Swiss with having strings and numerical values. The 3tr one is txhousing, with NA’s and good amount of float values and string values in it.

Here, I had to remove the cheractors and strings data, along with NA values.

Attitude <- select(attitude, rating:advance)  
Swiss <- select(swiss,Fertility:Infant.Mortality)  
Txhousing <- select(txhousing, year:date)  
Txh <- drop\_na(Txhousing)

## Question 1: Compute the variance - covariance matrix

Here, we will find the Covariance of the Datasets. We have, some inbuilt function called cov() from “matlib” Library. As we have to operate from the covariance of this datasets, we have to save them in the objects.

at <- cov(Attitude)  
at

## rating complaints privileges learning raises critical  
## rating 148.17126 133.77931 63.46437 89.10460 74.68851 18.84253  
## complaints 133.77931 177.28276 90.95172 93.25517 92.64138 24.73103  
## privileges 63.46437 90.95172 149.70575 70.84598 56.67126 17.82529  
## learning 89.10460 93.25517 70.84598 137.75747 78.13908 13.46782  
## raises 74.68851 92.64138 56.67126 78.13908 108.10230 38.77356  
## critical 18.84253 24.73103 17.82529 13.46782 38.77356 97.90920  
## advance 19.42299 30.76552 43.21609 64.19770 61.42299 28.84598  
## advance  
## rating 19.42299  
## complaints 30.76552  
## privileges 43.21609  
## learning 64.19770  
## raises 61.42299  
## critical 28.84598  
## advance 105.85747

sw <- cov(Swiss)  
sw

## Fertility Agriculture Examination Education Catholic  
## Fertility 156.04250 100.169149 -64.366929 -79.729510 241.56320  
## Agriculture 100.16915 515.799417 -124.392831 -139.657401 379.90438  
## Examination -64.36693 -124.392831 63.646623 53.575856 -190.56061  
## Education -79.72951 -139.657401 53.575856 92.456059 -61.69883  
## Catholic 241.56320 379.904376 -190.560611 -61.698830 1739.29454  
## Infant.Mortality 15.15619 -4.025851 -2.649537 -2.781684 21.31812  
## Infant.Mortality  
## Fertility 15.156193  
## Agriculture -4.025851  
## Examination -2.649537  
## Education -2.781684  
## Catholic 21.318116  
## Infant.Mortality 8.483802

tx <- cov(Txh)  
tx

## year month sales volume median  
## year 2.069070e+01 -7.034128e-01 -8.579315e+01 5.845779e+07 8.295102e+04  
## month -7.034128e-01 1.187101e+01 6.856229e+01 1.606513e+07 4.813997e+03  
## sales -8.579315e+01 6.856229e+01 1.356055e+06 2.937783e+11 1.473522e+07  
## volume 5.845779e+07 1.606513e+07 2.937783e+11 6.615967e+16 3.905877e+12  
## median 8.295102e+04 4.813997e+03 1.473522e+07 3.905877e+12 1.426113e+09  
## listings -1.471926e+03 1.232175e+02 6.422882e+06 1.325321e+12 5.552101e+07  
## inventory 2.048009e+00 2.307855e-01 -1.036576e+03 -2.305779e+08 -2.477815e+04  
## date 2.063208e+01 2.858376e-01 -8.007963e+01 5.979655e+07 8.335219e+04  
## listings inventory date  
## year -1.471926e+03 2.048009e+00 2.063208e+01  
## month 1.232175e+02 2.307855e-01 2.858376e-01  
## sales 6.422882e+06 -1.036576e+03 -8.007963e+01  
## volume 1.325321e+12 -2.305779e+08 5.979655e+07  
## median 5.552101e+07 -2.477815e+04 8.335219e+04  
## listings 3.583393e+07 -2.767827e+03 -1.461658e+03  
## inventory -2.767827e+03 2.129365e+01 2.067241e+00  
## date -1.461658e+03 2.067241e+00 2.065590e+01

# Now, as per the question 1 we have to calculate the variance too. Which is here.

var(Attitude)

## rating complaints privileges learning raises critical  
## rating 148.17126 133.77931 63.46437 89.10460 74.68851 18.84253  
## complaints 133.77931 177.28276 90.95172 93.25517 92.64138 24.73103  
## privileges 63.46437 90.95172 149.70575 70.84598 56.67126 17.82529  
## learning 89.10460 93.25517 70.84598 137.75747 78.13908 13.46782  
## raises 74.68851 92.64138 56.67126 78.13908 108.10230 38.77356  
## critical 18.84253 24.73103 17.82529 13.46782 38.77356 97.90920  
## advance 19.42299 30.76552 43.21609 64.19770 61.42299 28.84598  
## advance  
## rating 19.42299  
## complaints 30.76552  
## privileges 43.21609  
## learning 64.19770  
## raises 61.42299  
## critical 28.84598  
## advance 105.85747

var(Swiss)

## Fertility Agriculture Examination Education Catholic  
## Fertility 156.04250 100.169149 -64.366929 -79.729510 241.56320  
## Agriculture 100.16915 515.799417 -124.392831 -139.657401 379.90438  
## Examination -64.36693 -124.392831 63.646623 53.575856 -190.56061  
## Education -79.72951 -139.657401 53.575856 92.456059 -61.69883  
## Catholic 241.56320 379.904376 -190.560611 -61.698830 1739.29454  
## Infant.Mortality 15.15619 -4.025851 -2.649537 -2.781684 21.31812  
## Infant.Mortality  
## Fertility 15.156193  
## Agriculture -4.025851  
## Examination -2.649537  
## Education -2.781684  
## Catholic 21.318116  
## Infant.Mortality 8.483802

var(Txh)

## year month sales volume median  
## year 2.069070e+01 -7.034128e-01 -8.579315e+01 5.845779e+07 8.295102e+04  
## month -7.034128e-01 1.187101e+01 6.856229e+01 1.606513e+07 4.813997e+03  
## sales -8.579315e+01 6.856229e+01 1.356055e+06 2.937783e+11 1.473522e+07  
## volume 5.845779e+07 1.606513e+07 2.937783e+11 6.615967e+16 3.905877e+12  
## median 8.295102e+04 4.813997e+03 1.473522e+07 3.905877e+12 1.426113e+09  
## listings -1.471926e+03 1.232175e+02 6.422882e+06 1.325321e+12 5.552101e+07  
## inventory 2.048009e+00 2.307855e-01 -1.036576e+03 -2.305779e+08 -2.477815e+04  
## date 2.063208e+01 2.858376e-01 -8.007963e+01 5.979655e+07 8.335219e+04  
## listings inventory date  
## year -1.471926e+03 2.048009e+00 2.063208e+01  
## month 1.232175e+02 2.307855e-01 2.858376e-01  
## sales 6.422882e+06 -1.036576e+03 -8.007963e+01  
## volume 1.325321e+12 -2.305779e+08 5.979655e+07  
## median 5.552101e+07 -2.477815e+04 8.335219e+04  
## listings 3.583393e+07 -2.767827e+03 -1.461658e+03  
## inventory -2.767827e+03 2.129365e+01 2.067241e+00  
## date -1.461658e+03 2.067241e+00 2.065590e+01

## Questions 2: Check to see if the matrix is square, symmetric, orthogonal?

For the **Square matrix, it has to have same numbers of Columns and Rows, then nd then only it will be called as square matrix.** As you can see the result of the same from the above.

is\_square\_matrix(at)

## [1] TRUE

is\_square\_matrix(sw)

## [1] TRUE

is\_square\_matrix(tx)

## [1] TRUE

As per the definition, **A matrix is the Symmatric if it is a Square matrix and it’s transpose is as same as it’s original matrix.** You can see that result over here what we calculated. And just to make sure, we also calculated it with inbuilt function. That also cross checks with our results.

Trans\_at <- t(at)  
Trans\_at

## rating complaints privileges learning raises critical  
## rating 148.17126 133.77931 63.46437 89.10460 74.68851 18.84253  
## complaints 133.77931 177.28276 90.95172 93.25517 92.64138 24.73103  
## privileges 63.46437 90.95172 149.70575 70.84598 56.67126 17.82529  
## learning 89.10460 93.25517 70.84598 137.75747 78.13908 13.46782  
## raises 74.68851 92.64138 56.67126 78.13908 108.10230 38.77356  
## critical 18.84253 24.73103 17.82529 13.46782 38.77356 97.90920  
## advance 19.42299 30.76552 43.21609 64.19770 61.42299 28.84598  
## advance  
## rating 19.42299  
## complaints 30.76552  
## privileges 43.21609  
## learning 64.19770  
## raises 61.42299  
## critical 28.84598  
## advance 105.85747

at

## rating complaints privileges learning raises critical  
## rating 148.17126 133.77931 63.46437 89.10460 74.68851 18.84253  
## complaints 133.77931 177.28276 90.95172 93.25517 92.64138 24.73103  
## privileges 63.46437 90.95172 149.70575 70.84598 56.67126 17.82529  
## learning 89.10460 93.25517 70.84598 137.75747 78.13908 13.46782  
## raises 74.68851 92.64138 56.67126 78.13908 108.10230 38.77356  
## critical 18.84253 24.73103 17.82529 13.46782 38.77356 97.90920  
## advance 19.42299 30.76552 43.21609 64.19770 61.42299 28.84598  
## advance  
## rating 19.42299  
## complaints 30.76552  
## privileges 43.21609  
## learning 64.19770  
## raises 61.42299  
## critical 28.84598  
## advance 105.85747

is\_symmetric\_matrix(at)

## [1] TRUE

Trans\_sw <- t(sw)  
Trans\_sw

## Fertility Agriculture Examination Education Catholic  
## Fertility 156.04250 100.169149 -64.366929 -79.729510 241.56320  
## Agriculture 100.16915 515.799417 -124.392831 -139.657401 379.90438  
## Examination -64.36693 -124.392831 63.646623 53.575856 -190.56061  
## Education -79.72951 -139.657401 53.575856 92.456059 -61.69883  
## Catholic 241.56320 379.904376 -190.560611 -61.698830 1739.29454  
## Infant.Mortality 15.15619 -4.025851 -2.649537 -2.781684 21.31812  
## Infant.Mortality  
## Fertility 15.156193  
## Agriculture -4.025851  
## Examination -2.649537  
## Education -2.781684  
## Catholic 21.318116  
## Infant.Mortality 8.483802

sw

## Fertility Agriculture Examination Education Catholic  
## Fertility 156.04250 100.169149 -64.366929 -79.729510 241.56320  
## Agriculture 100.16915 515.799417 -124.392831 -139.657401 379.90438  
## Examination -64.36693 -124.392831 63.646623 53.575856 -190.56061  
## Education -79.72951 -139.657401 53.575856 92.456059 -61.69883  
## Catholic 241.56320 379.904376 -190.560611 -61.698830 1739.29454  
## Infant.Mortality 15.15619 -4.025851 -2.649537 -2.781684 21.31812  
## Infant.Mortality  
## Fertility 15.156193  
## Agriculture -4.025851  
## Examination -2.649537  
## Education -2.781684  
## Catholic 21.318116  
## Infant.Mortality 8.483802

is\_symmetric\_matrix(sw)

## [1] TRUE

Trans\_tx <- t(tx)  
Trans\_tx

## year month sales volume median  
## year 2.069070e+01 -7.034128e-01 -8.579315e+01 5.845779e+07 8.295102e+04  
## month -7.034128e-01 1.187101e+01 6.856229e+01 1.606513e+07 4.813997e+03  
## sales -8.579315e+01 6.856229e+01 1.356055e+06 2.937783e+11 1.473522e+07  
## volume 5.845779e+07 1.606513e+07 2.937783e+11 6.615967e+16 3.905877e+12  
## median 8.295102e+04 4.813997e+03 1.473522e+07 3.905877e+12 1.426113e+09  
## listings -1.471926e+03 1.232175e+02 6.422882e+06 1.325321e+12 5.552101e+07  
## inventory 2.048009e+00 2.307855e-01 -1.036576e+03 -2.305779e+08 -2.477815e+04  
## date 2.063208e+01 2.858376e-01 -8.007963e+01 5.979655e+07 8.335219e+04  
## listings inventory date  
## year -1.471926e+03 2.048009e+00 2.063208e+01  
## month 1.232175e+02 2.307855e-01 2.858376e-01  
## sales 6.422882e+06 -1.036576e+03 -8.007963e+01  
## volume 1.325321e+12 -2.305779e+08 5.979655e+07  
## median 5.552101e+07 -2.477815e+04 8.335219e+04  
## listings 3.583393e+07 -2.767827e+03 -1.461658e+03  
## inventory -2.767827e+03 2.129365e+01 2.067241e+00  
## date -1.461658e+03 2.067241e+00 2.065590e+01

tx

## year month sales volume median  
## year 2.069070e+01 -7.034128e-01 -8.579315e+01 5.845779e+07 8.295102e+04  
## month -7.034128e-01 1.187101e+01 6.856229e+01 1.606513e+07 4.813997e+03  
## sales -8.579315e+01 6.856229e+01 1.356055e+06 2.937783e+11 1.473522e+07  
## volume 5.845779e+07 1.606513e+07 2.937783e+11 6.615967e+16 3.905877e+12  
## median 8.295102e+04 4.813997e+03 1.473522e+07 3.905877e+12 1.426113e+09  
## listings -1.471926e+03 1.232175e+02 6.422882e+06 1.325321e+12 5.552101e+07  
## inventory 2.048009e+00 2.307855e-01 -1.036576e+03 -2.305779e+08 -2.477815e+04  
## date 2.063208e+01 2.858376e-01 -8.007963e+01 5.979655e+07 8.335219e+04  
## listings inventory date  
## year -1.471926e+03 2.048009e+00 2.063208e+01  
## month 1.232175e+02 2.307855e-01 2.858376e-01  
## sales 6.422882e+06 -1.036576e+03 -8.007963e+01  
## volume 1.325321e+12 -2.305779e+08 5.979655e+07  
## median 5.552101e+07 -2.477815e+04 8.335219e+04  
## listings 3.583393e+07 -2.767827e+03 -1.461658e+03  
## inventory -2.767827e+03 2.129365e+01 2.067241e+00  
## date -1.461658e+03 2.067241e+00 2.065590e+01

is\_symmetric\_matrix(tx)

## [1] TRUE

To test whether a matrix is an **orthogonal matrix, we multiply the matrix to its transpose. If the result is an identity matrix, then the input matrix is an orthogonal matrix.**

Even though we have calculated the Transpose of the matrix and it is not an Identity matrix, we can always double ckeck our ans with inbuilt function from “matlib”.

resA <- Trans\_at %\*% at  
resA

## rating complaints privileges learning raises critical advance  
## rating 58129.65 65603.47 42792.69 49786.14 44016.96 13732.81 22643.90  
## complaints 65603.47 76435.39 51857.65 57290.11 51720.09 16683.20 27630.40  
## privileges 42792.69 51857.65 45127.91 41944.98 36657.71 12257.06 23618.54  
## learning 49786.14 57290.11 41944.98 51040.87 42985.97 13303.59 28488.95  
## raises 44016.96 51720.09 36657.71 42985.97 40440.43 15520.59 26026.80  
## critical 13732.81 16683.20 12257.06 13303.59 15520.59 13387.48 11021.22  
## advance 22643.90 27630.40 23618.54 28488.95 26026.80 11021.22 23123.42

is\_orthogonal\_matrix(at)

## [1] FALSE

resS <- Trans\_sw %\*% sw  
resS

## Fertility Agriculture Examination Education Catholic  
## Fertility 103465.507 178149.335 -76945.203 -52196.871 513406.50  
## Agriculture 178149.335 455404.205 -158392.922 -123026.725 913152.49  
## Examination -76945.203 -158392.922 62858.309 42632.395 -409737.68  
## Education -52196.871 -123026.725 42632.395 41093.963 -195601.79  
## Catholic 513406.503 913152.489 -409737.681 -195601.790 3268400.16  
## Infant.Mortality 7632.324 8224.409 -4877.307 -2384.193 40067.61  
## Infant.Mortality  
## Fertility 7632.3236  
## Agriculture 8224.4086  
## Examination -4877.3072  
## Education -2384.1928  
## Catholic 40067.6058  
## Infant.Mortality 787.1125

is\_orthogonal\_matrix(sw)

## [1] FALSE

resT <- Trans\_tx %\*% tx  
resT

## year month sales volume median  
## year 3.417320e+15 9.391321e+14 1.717363e+19 3.867548e+24 2.283290e+20  
## month 9.391321e+14 2.580883e+14 4.719586e+18 1.062864e+24 6.274842e+19  
## sales 1.717363e+19 4.719586e+18 8.630571e+22 1.943628e+28 1.147462e+24  
## volume 3.867548e+24 1.062864e+24 1.943628e+28 4.377102e+33 2.584116e+29  
## median 2.283290e+20 6.274842e+19 1.147462e+24 2.584116e+29 1.525588e+25  
## listings 7.747531e+19 2.129144e+19 3.893505e+23 8.768277e+28 5.176539e+24  
## inventory -1.347908e+16 -3.704264e+15 -6.773880e+19 -1.525496e+25 -9.006091e+20  
## date 3.495581e+15 9.606395e+14 1.756693e+19 3.956120e+24 2.335581e+20  
## listings inventory date  
## year 7.747531e+19 -1.347908e+16 3.495581e+15  
## month 2.129144e+19 -3.704264e+15 9.606395e+14  
## sales 3.893505e+23 -6.773880e+19 1.756693e+19  
## volume 8.768277e+28 -1.525496e+25 3.956120e+24  
## median 5.176539e+24 -9.006091e+20 2.335581e+20  
## listings 1.756475e+24 -3.055897e+20 7.924960e+19  
## inventory -3.055897e+20 5.316618e+16 -1.378777e+16  
## date 7.924960e+19 -1.378777e+16 3.575634e+15

is\_orthogonal\_matrix(tx)

## [1] FALSE

## 

## Question 3: Is it positive definite?

Definition says that a **matrix is positive definite if it’s symmetric and all its eigenvalues are positive. As you can see from the results that we have all positive eigen values.** so they all are Positive definite matrixs.

evAt <- eigen(at)  
eigen.values.At <- evAt$values  
eigen.values.At

## [1] 519.79278 134.14087 97.06354 85.23927 41.01460 25.74079 21.79436

evSw <- eigen(sw)  
eigen.values.Sw <- evSw$values  
eigen.values.Sw

## [1] 1921.562488 466.657132 145.284829 22.649607 13.377631 6.191249

evTx <- eigen(tx)  
eigen.values.Tx <- evTx$values  
eigen.values.Tx

## [1] 6.615967e+16 1.195962e+09 8.876783e+06 1.886930e+04 2.973952e+01  
## [6] 1.829398e+01 1.091698e+01 2.598971e-01

## 

## Question 4: Compute the eigenvalues and eigenvectors.

As we have calculated the eigenvalues before, here I am showing you only eigenvalues and eigenvectors. And we going to store them for the further calculations.

eigen.values.At <- evAt$values  
eigen.vector.At <- evAt$vectors  
eigen.values.At

## [1] 519.79278 134.14087 97.06354 85.23927 41.01460 25.74079 21.79436

eigen.vector.At

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -0.4467200 0.42184464 -0.2400295 0.1261907 0.20102006 0.47248738  
## [2,] -0.5206244 0.37207702 -0.1432265 -0.1081087 -0.37237853 0.02200456  
## [3,] -0.3757726 -0.07632652 0.6513217 -0.6263275 0.07783612 -0.01814123  
## [4,] -0.4209952 -0.14566925 0.1864780 0.4851365 0.62078169 -0.30156007  
## [5,] -0.3762536 -0.23339685 -0.2239163 0.1041004 -0.44682483 -0.59329347  
## [6,] -0.1300302 -0.39828981 -0.6330345 -0.5170667 0.37798712 0.01806699  
## [7,] -0.2290738 -0.66592166 0.1095758 0.2579729 -0.29490695 0.57678447  
## [,7]  
## [1,] 0.5341317  
## [2,] -0.6474239  
## [3,] 0.1734231  
## [4,] -0.2347416  
## [5,] 0.4374176  
## [6,] -0.1147434  
## [7,] -0.0765914

eigen.values.Sw <- evSw$values  
eigen.vector.Sw <- evSw$vectors  
eigen.values.Sw

## [1] 1921.562488 466.657132 145.284829 22.649607 13.377631 6.191249

eigen.vector.Sw

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -0.15163143 -0.14270789 0.81454413 -0.49552828 0.12247267 -0.1805890073  
## [2,] -0.28121756 -0.85914886 -0.35256541 -0.24078519 0.02235164 0.0006755062  
## [3,] 0.12207834 0.17688621 -0.18767793 -0.57042350 -0.76887882 -0.0450281625  
## [4,] 0.06329733 0.32260928 -0.40096045 -0.58075837 0.61904343 -0.1032102386  
## [5,] -0.93748965 0.32543441 -0.07870742 0.04104832 -0.08547871 0.0043788222  
## [6,] -0.01131739 0.01498883 0.10014161 -0.17923725 0.05296062 0.9770814149

eigen.values.Tx <- evTx$values  
eigen.vector.Tx <- evTx$vectors  
eigen.values.Tx

## [1] 6.615967e+16 1.195962e+09 8.876783e+06 1.886930e+04 2.973952e+01  
## [6] 1.829398e+01 1.091698e+01 2.598971e-01

eigen.vector.Tx

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 8.835865e-10 6.650415e-05 1.270935e-04 5.795609e-03 -6.967810e-01  
## [2,] 2.428234e-10 3.234761e-06 1.397517e-05 -6.630283e-04 4.469868e-02  
## [3,] 4.440444e-06 -2.189483e-03 -5.515661e-02 -9.984189e-01 -8.925397e-03  
## [4,] 1.000000e+00 -5.863293e-05 2.137855e-05 3.394686e-06 2.907938e-08  
## [5,] 5.903713e-05 9.998143e-01 -1.923678e-02 -1.130400e-03 7.908123e-05  
## [6,] 2.003215e-05 -1.914512e-02 -9.982924e-01 5.518559e-02 3.349401e-04  
## [7,] -3.485173e-09 -9.363870e-06 -1.839065e-04 6.785873e-03 -1.198542e-01  
## [8,] 9.038218e-10 6.677371e-05 1.282581e-04 5.736560e-03 -7.057290e-01  
## [,6] [,7] [,8]  
## [1,] 0.000000e+00 0.000000e+00 7.172605e-01  
## [2,] -1.343159e-01 9.889767e-01 4.342778e-02  
## [3,] -5.625544e-03 -1.003147e-03 -5.931531e-04  
## [4,] 1.253528e-08 2.253901e-09 1.235751e-09  
## [5,] -2.181699e-05 -1.014695e-05 -3.336693e-06  
## [6,] 5.098408e-04 1.027190e-04 5.813155e-05  
## [7,] -9.782922e-01 -1.223280e-01 -1.164869e-01  
## [8,] 1.577080e-01 8.342645e-02 -6.856251e-01

## 

## Question 5: Find inverse and square root of the matrix using spectral decomposition.

I tried to explain how I solved this spectral decomposition, to find the Inverse. And just like the same way I calculated the Square root of the Matrix, but different Equation. **As you have proposed the same in Liner2.pdf on page 18 to 20.**

**suppose A = PDP’**

**where A = our data orthogonal matrix**

**P = Matrix of eigenvector of A**

**P’ = Eigenvector transpose or A**

**D = Diagonal matrix with eigenvalues or A**

Let’s find the Transposition of the eigenvectors.

trans.eigen.vector.At <- t(eigen.vector.At)  
trans.eigen.vector.At

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -0.4467200 -0.52062436 -0.37577262 -0.4209952 -0.3762536 -0.13003020  
## [2,] 0.4218446 0.37207702 -0.07632652 -0.1456692 -0.2333969 -0.39828981  
## [3,] -0.2400295 -0.14322649 0.65132166 0.1864780 -0.2239163 -0.63303451  
## [4,] 0.1261907 -0.10810869 -0.62632750 0.4851365 0.1041004 -0.51706669  
## [5,] 0.2010201 -0.37237853 0.07783612 0.6207817 -0.4468248 0.37798712  
## [6,] 0.4724874 0.02200456 -0.01814123 -0.3015601 -0.5932935 0.01806699  
## [7,] 0.5341317 -0.64742389 0.17342314 -0.2347416 0.4374176 -0.11474339  
## [,7]  
## [1,] -0.2290738  
## [2,] -0.6659217  
## [3,] 0.1095758  
## [4,] 0.2579729  
## [5,] -0.2949069  
## [6,] 0.5767845  
## [7,] -0.0765914

trans.eigen.vector.Sw <- t(eigen.vector.Sw)  
trans.eigen.vector.Sw

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -0.1516314 -0.2812175593 0.12207834 0.06329733 -0.937489645 -0.01131739  
## [2,] -0.1427079 -0.8591488560 0.17688621 0.32260928 0.325434415 0.01498883  
## [3,] 0.8145441 -0.3525654108 -0.18767793 -0.40096045 -0.078707418 0.10014161  
## [4,] -0.4955283 -0.2407851877 -0.57042350 -0.58075837 0.041048316 -0.17923725  
## [5,] 0.1224727 0.0223516380 -0.76887882 0.61904343 -0.085478715 0.05296062  
## [6,] -0.1805890 0.0006755062 -0.04502816 -0.10321024 0.004378822 0.97708141

trans.eigen.vector.Tx <- t(eigen.vector.Tx)  
trans.eigen.vector.Tx

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 8.835865e-10 2.428234e-10 4.440444e-06 1.000000e+00 5.903713e-05  
## [2,] 6.650415e-05 3.234761e-06 -2.189483e-03 -5.863293e-05 9.998143e-01  
## [3,] 1.270935e-04 1.397517e-05 -5.515661e-02 2.137855e-05 -1.923678e-02  
## [4,] 5.795609e-03 -6.630283e-04 -9.984189e-01 3.394686e-06 -1.130400e-03  
## [5,] -6.967810e-01 4.469868e-02 -8.925397e-03 2.907938e-08 7.908123e-05  
## [6,] 0.000000e+00 -1.343159e-01 -5.625544e-03 1.253528e-08 -2.181699e-05  
## [7,] 0.000000e+00 9.889767e-01 -1.003147e-03 2.253901e-09 -1.014695e-05  
## [8,] 7.172605e-01 4.342778e-02 -5.931531e-04 1.235751e-09 -3.336693e-06  
## [,6] [,7] [,8]  
## [1,] 2.003215e-05 -3.485173e-09 9.038218e-10  
## [2,] -1.914512e-02 -9.363870e-06 6.677371e-05  
## [3,] -9.982924e-01 -1.839065e-04 1.282581e-04  
## [4,] 5.518559e-02 6.785873e-03 5.736560e-03  
## [5,] 3.349401e-04 -1.198542e-01 -7.057290e-01  
## [6,] 5.098408e-04 -9.782922e-01 1.577080e-01  
## [7,] 1.027190e-04 -1.223280e-01 8.342645e-02  
## [8,] 5.813155e-05 -1.164869e-01 -6.856251e-01

### Now, we will compute Inverse of the Matrix as Spectral decomposition. For which we have to calculate the 3 different Matrix multiplications.

inv.At <- eigen.vector.At %\*% inv(diag(eigen.values.At)) %\*% trans.eigen.vector.At  
inv.At

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.0252393311 -0.0154764468 0.0018437365 -0.0080849698 -0.002062864  
## [2,] -0.0154764468 0.0245340687 -0.0058759670 0.0002064222 -0.009516478  
## [3,] 0.0018437365 -0.0058759670 0.0108282711 -0.0024034413 0.001188159  
## [4,] -0.0080849698 0.0002064222 -0.0024034413 0.0190756804 -0.003803213  
## [5,] -0.0020628636 -0.0095164784 0.0011881588 -0.0038032129 0.028643724  
## [6,] -0.0009687227 0.0006075736 -0.0003362994 0.0031240694 -0.005221254  
## [7,] 0.0054783645 0.0033392473 -0.0021913746 -0.0078083878 -0.010231822  
## [,6] [,7]  
## [1,] -0.0009687227 0.005478364  
## [2,] 0.0006075736 0.003339247  
## [3,] -0.0003362994 -0.002191375  
## [4,] 0.0031240694 -0.007808388  
## [5,] -0.0052212539 -0.010231822  
## [6,] 0.0125805267 -0.002154727  
## [7,] -0.0021547267 0.019625138

inv.Sw <- eigen.vector.Sw %\*% inv(diag(eigen.values.Sw)) %\*% trans.eigen.vector.Sw  
inv.Sw

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.021852286 0.0037610843 0.005638071 0.019032033 -2.275158e-03  
## [2,] 0.003761084 0.0050756632 0.004886446 0.007566830 -8.496683e-04  
## [3,] 0.005638071 0.0048864458 0.059201954 -0.019558373 4.012720e-03  
## [4,] 0.019032033 0.0075668300 -0.019558373 0.046589399 -4.669686e-03  
## [5,] -0.002275158 -0.0008496683 0.004012720 -0.004669686 1.350641e-03  
## [6,] -0.023535963 0.0018315884 -0.005760461 -0.009508117 -1.046268e-05  
## [,6]  
## [1,] -2.353596e-02  
## [2,] 1.831588e-03  
## [3,] -5.760461e-03  
## [4,] -9.508117e-03  
## [5,] -1.046268e-05  
## [6,] 1.558972e-01

inv.Tx <- eigen.vector.Tx %\*% inv(diag(eigen.values.Tx)) %\*% trans.eigen.vector.Tx  
inv.Tx

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 1.995811e+00 1.188041e-01 -1.428165e-03 2.730137e-09 -1.106174e-05  
## [2,] 1.188041e-01 9.790205e-02 -1.620660e-04 3.622237e-10 -1.197686e-06  
## [3,] -1.428165e-03 -1.620660e-04 5.868736e-05 -1.953730e-10 5.145587e-08  
## [4,] 2.730137e-09 3.622237e-10 -1.953730e-10 7.044054e-16 -2.042012e-13  
## [5,] -1.106174e-05 -1.197686e-06 5.145587e-08 -2.042012e-13 3.970045e-10  
## [6,] 1.526001e-04 1.577712e-05 -3.313567e-06 8.555743e-12 -1.752974e-09  
## [7,] -3.186708e-01 -2.354372e-02 6.135382e-04 -1.365436e-09 2.456794e-06  
## [8,] -1.875644e+00 -1.092262e-01 1.720112e-03 -3.823733e-09 6.659819e-06  
## [,6] [,7] [,8]  
## [1,] 1.526001e-04 -3.186708e-01 -1.875644e+00  
## [2,] 1.577712e-05 -2.354372e-02 -1.092262e-01  
## [3,] -3.313567e-06 6.135382e-04 1.720112e-03  
## [4,] 8.555743e-12 -1.365436e-09 -3.823733e-09  
## [5,] -1.752974e-09 2.456794e-06 6.659819e-06  
## [6,] 3.029835e-07 -5.580011e-05 -1.561060e-04  
## [7,] -5.580011e-05 1.063790e-01 3.007756e-01  
## [8,] -1.561060e-04 3.007756e-01 1.827467e+00

### The Exact same way but with the different Equation we will calculate the square root of the Matrix.

Sqr.At <- eigen.vector.At %\*% diag(sqrt(eigen.values.At)) %\*% trans.eigen.vector.At  
Sqr.At

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 10.04872876 5.2920237 1.6734498 3.19120419 2.435785 0.51676012  
## [2,] 5.29202369 10.9404003 3.1256538 2.81737072 3.349562 0.68379290  
## [3,] 1.67344978 3.1256538 11.2689144 2.27395576 1.576976 0.48782276  
## [4,] 3.19120419 2.8173707 2.2739558 9.98875835 2.711989 0.04191152  
## [5,] 2.43578458 3.3495620 1.5769759 2.71198930 8.410242 1.72128036  
## [6,] 0.51676012 0.6837929 0.4878228 0.04191152 1.721280 9.61733950  
## [7,] -0.06703763 0.4364140 1.5004902 2.70800638 2.722678 1.21607111  
## [,7]  
## [1,] -0.06703763  
## [2,] 0.43641398  
## [3,] 1.50049021  
## [4,] 2.70800638  
## [5,] 2.72267844  
## [6,] 1.21607111  
## [7,] 9.33734137

Sqr.Sw <- eigen.vector.Sw %\*% diag(sqrt(eigen.values.Sw)) %\*% trans.eigen.vector.Sw  
Sqr.Sw

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 10.749652 1.6338586 -2.178330159 -3.65863863 4.3182946 1.019583014  
## [2,] 1.633859 21.1881013 -3.399535548 -4.34789571 5.7973037 -0.352869815  
## [3,] -2.178330 -3.3995355 5.469594384 2.32579073 -3.4668273 -0.001652185  
## [4,] -3.658639 -4.3478957 2.325790726 7.39504198 -0.2609815 -0.046536506  
## [5,] 4.318295 5.7973037 -3.466827268 -0.26098154 40.9238600 0.434535719  
## [6,] 1.019583 -0.3528698 -0.001652185 -0.04653651 0.4345357 2.669971112

Sqr.Tx <- eigen.vector.Tx %\*% diag(sqrt(eigen.values.Tx)) %\*% trans.eigen.vector.Tx  
Sqr.Tx

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 2.9147326 -0.154482337 -0.7870807 2.271477e-01 2.2909924  
## [2,] -0.1544823 3.320723816 0.0861571 6.245191e-02 0.1111502  
## [3,] -0.7870807 0.086157098 146.1670296 1.142150e+03 -72.3204561  
## [4,] 0.2271477 0.062451912 1142.1503464 2.572152e+08 15183.2202198  
## [5,] 2.2909924 0.111150203 -72.3204561 1.518322e+04 34571.8418242  
## [6,] -0.3793593 -0.048607349 157.9564569 5.152550e+03 -604.4548933  
## [7,] 0.4181413 0.129869587 -0.8699311 -8.964293e-01 -0.3142899  
## [8,] 2.4357082 -0.005709181 -0.7824053 2.323521e-01 2.3002366  
## [,6] [,7] [,8]  
## [1,] -0.37935927 0.4181413 2.435708155  
## [2,] -0.04860735 0.1298696 -0.005709181  
## [3,] 157.95645687 -0.8699311 -0.782405294  
## [4,] 5152.54954101 -0.8964293 0.232352063  
## [5,] -604.45489334 -0.3142899 2.300236644  
## [6,] 2982.42364510 0.6022197 -0.383134841  
## [7,] 0.60221971 4.2345940 -0.186371989  
## [8,] -0.38313484 -0.1863720 3.089830943

**Conclusion**: As far as I learned from this Assignment, I can easily say that Eigenvectors and Eigenvalues simplifies the Matrix operation for us. And save good amount of time and storage for the calculation. They shows the direction of the variance, in which side data is distributed. The number of Eigenvalues/Vector somehow propose the numbers of directions that data has, from which all we can get some kind of conclusions depending on the data.

However, All the data set are in this project are not related with each other, and haven’t work on connected data set to drew result after this matrix operations.