**Singular Value Decomposition (SVD) - un esempio con R**

**PADEP**

2019-05-05

library(dslabs)  
library(caret)

library(tidyverse)

data("swiss")  
str(swiss)

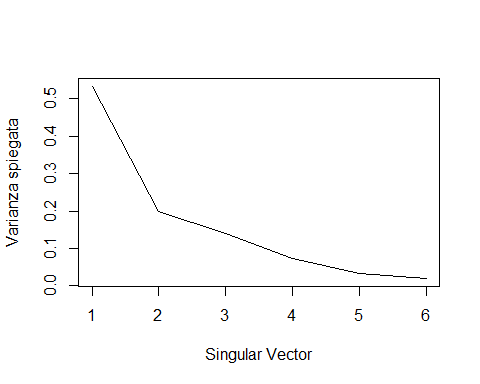
## 'data.frame': 47 obs. of 6 variables:  
## $ Fertility : num 80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4 82.9 ...  
## $ Agriculture : num 17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3 45.2 ...  
## $ Examination : int 15 6 5 12 17 9 16 14 12 16 ...  
## $ Education : int 12 9 5 7 15 7 7 8 7 13 ...  
## $ Catholic : num 9.96 84.84 93.4 33.77 5.16 ...  
## $ Infant.Mortality: num 22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21 24.4 ...

dim(swiss)

## [1] 47 6

write.table(swiss, file="swiss.csv", quote=T, sep=";", dec=",", na="NA", row.names=T, col.names=T)  
  
# Visto che le colonne della amtrice originale si riferiscono a diverse unità di misura, Standardiziamo la Matrice di partenza  
# Questo dovrebbe ridurre anche l'indice RMSE  
swiss <- scale(swiss) # commentare questa riga e vedere le differenze  
  
swiss.svd = svd(swiss)  
class(swiss.svd) #list

plot(swiss.svd$d^2/sum(swiss.svd$d^2), type="l", xlab="Singular Vector", ylab = "Varianza spiegata")



swiss.svd$d

## [1] 12.132140 7.393388 6.244201 4.493408 3.067336 2.357897

swiss.svd$u

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.02964549 0.18725556 -0.13620837 0.2005649858 0.20371260  
## [2,] -0.13325738 0.13729469 0.08680173 0.1113877876 0.20846924  
## [3,] -0.17158512 0.09982495 0.07488988 0.3306727249 0.13148026  
## [4,] -0.06096302 0.07973588 -0.09175955 0.2360857999 0.07415829  
## [5,] 0.03111133 0.06005964 -0.09953902 0.0541783577 0.02294084  
## [6,] -0.11165174 0.30667479 0.05553606 -0.0677638891 0.26980810  
## [7,] -0.14060873 0.15090383 0.06847160 -0.1028435040 -0.26632013  
## [8,] -0.17797294 0.23605901 0.06310325 -0.0549190876 -0.26137241  
## [9,] -0.12375699 0.08330601 0.11216815 0.1035503329 -0.06558003  
## [10,] -0.07840919 0.25358505 0.10532343 -0.0004346082 -0.09676269  
## [11,] -0.16400876 0.20624800 0.06429721 -0.0633209843 -0.20222200  
## [12,] 0.06564838 -0.18183673 -0.05344662 -0.0170122251 -0.14181211  
## [13,] -0.01428726 -0.11800306 -0.13196451 -0.1110055639 0.06078063  
## [14,] 0.01921377 0.06545231 -0.12807918 -0.2116980521 -0.03785403  
## [15,] 0.03427773 -0.14343387 -0.14705468 -0.1632149018 -0.17343781  
## [16,] -0.04883641 -0.04025811 -0.13591022 -0.1974658334 -0.13271300  
## [17,] 0.03798456 0.02409942 -0.15307470 0.1033534644 0.10474962  
## [18,] 0.22705340 0.08524083 0.06501703 -0.0194554864 0.08199128  
## [19,] 0.27962224 -0.27768813 0.03879647 0.3771332282 -0.16288792  
## [20,] 0.01258758 -0.08795298 -0.12404729 -0.2108553999 -0.10264653  
## [21,] 0.05821490 -0.11953602 -0.10012393 -0.0489248989 -0.15982719  
## [22,] -0.02057857 0.02105506 -0.18863291 -0.1736243652 0.17221889  
## [23,] 0.10519203 -0.16799955 -0.02282201 -0.0246974170 -0.04749710  
## [24,] 0.07745284 -0.25031377 -0.09484895 0.0298795001 -0.02141751  
## [25,] -0.07890697 -0.05526509 -0.20930303 -0.1407247188 0.05893894  
## [26,] -0.03988753 0.11144540 -0.17410561 -0.1719855095 0.09156553  
## [27,] -0.07675864 -0.15874525 -0.15464265 0.0284527436 0.27455205  
## [28,] 0.04563083 -0.21499522 -0.05504605 -0.0240716706 0.06098021  
## [29,] 0.16014488 0.08075549 -0.00323662 -0.0442656527 0.02673882  
## [30,] 0.01005900 0.05540088 -0.14451875 -0.1505035752 0.17610469  
## [31,] -0.19643479 -0.27359125 0.16676031 0.0958289919 0.04335850  
## [32,] -0.15944156 -0.09718179 0.16697733 -0.1613050564 0.04235037  
## [33,] -0.21093400 -0.15171467 0.13284333 -0.0426167929 -0.02461647  
## [34,] -0.12422015 -0.08138771 0.15462550 -0.0995244924 -0.10361443  
## [35,] -0.17147251 -0.00612928 0.10442298 0.0509497797 0.04852082  
## [36,] -0.10150284 -0.15395233 0.21274724 -0.0752653501 0.05198550  
## [37,] -0.24580556 -0.16457372 0.12673424 0.2126741186 -0.09979137  
## [38,] -0.08800014 -0.04392330 0.19602244 0.1134640686 -0.13880359  
## [39,] 0.09523028 0.05033127 -0.12545666 0.0296973368 -0.19554112  
## [40,] 0.16207278 0.12442005 -0.11409744 0.1722458788 -0.09491992  
## [41,] 0.10658738 0.05548727 -0.09058060 0.2514956309 0.00938767  
## [42,] 0.25137915 0.25688301 0.05438777 -0.0579382943 -0.25922362  
## [43,] -0.00093182 0.02988134 -0.16527535 0.1431463795 0.08812260  
## [44,] 0.11055633 0.04543468 -0.15062030 0.1736880276 -0.05404883  
## [45,] 0.46118557 0.07537994 0.42076922 -0.0707707187 0.01240803  
## [46,] 0.15153064 -0.09565798 0.26038716 -0.1819933415 0.30248007  
## [47,] 0.20783154 0.00192543 0.26331268 -0.1302477473 0.22510624  
## [,6]  
## [1,] 0.118893936  
## [2,] 0.038277679  
## [3,] 0.036588407  
## [4,] 0.130056317  
## [5,] 0.321942904  
## [6,] -0.297616257  
## [7,] -0.032899014  
## [8,] 0.113460466  
## [9,] -0.096647152  
## [10,] -0.013968893  
## [11,] -0.052311118  
## [12,] 0.116112771  
## [13,] 0.123326099  
## [14,] 0.158914216  
## [15,] -0.103029504  
## [16,] -0.135071272  
## [17,] -0.012397684  
## [18,] 0.101712302  
## [19,] -0.101942587  
## [20,] 0.114537548  
## [21,] 0.051112279  
## [22,] -0.143816495  
## [23,] -0.109461262  
## [24,] -0.174575950  
## [25,] 0.057057620  
## [26,] 0.153242074  
## [27,] 0.168074031  
## [28,] 0.056319195  
## [29,] -0.105803312  
## [30,] -0.038988738  
## [31,] -0.006755740  
## [32,] -0.090510343  
## [33,] -0.015699306  
## [34,] -0.162120275  
## [35,] -0.132082625  
## [36,] -0.120916196  
## [37,] 0.271927973  
## [38,] 0.113258955  
## [39,] -0.007718585  
## [40,] -0.361505001  
## [41,] -0.025917135  
## [42,] 0.193892246  
## [43,] 0.090051701  
## [44,] -0.308183109  
## [45,] 0.184379271  
## [46,] 0.095984572  
## [47,] -0.159185006

swiss.svd$v

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -0.4569876 0.3220284 -0.17376638 0.53555794 -0.38308893 0.47295441  
## [2,] -0.4242141 -0.4115132 0.03834472 -0.64291822 -0.37495215 0.30870058  
## [3,] 0.5097327 0.1250167 -0.09123696 -0.05446158 -0.81429082 -0.22401686  
## [4,] 0.4543119 0.1790495 0.53239316 -0.09738818 0.07144564 0.68081610  
## [5,] -0.3501111 0.1458730 0.80680494 0.09947244 -0.18317236 -0.40219666  
## [6,] -0.1496668 0.8111645 -0.16010636 -0.52677184 0.10453530 -0.07457754

sum(swiss.svd$d \* swiss.svd$u[5, ] \* swiss.svd$v[2, ]) # valore per la 5 riga 2 colonna matrice originale

## [1] -0.315244

sum(swiss.svd$d \* swiss.svd$u[6, ] \* swiss.svd$v[1, ]) # valore per la 6 riga 1 colonna matrice originale

## [1] 0.4769125

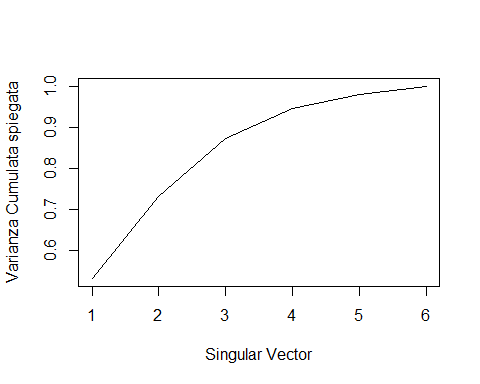
swiss.svd$u %\*% diag(swiss.svd$d) %\*% t(swiss.svd$v) # ricrea la matrice originale

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.80513053 -1.48206823 -0.18668632 0.106212514 -0.7477267  
## [2,] 1.03728474 -0.24479420 -1.31480509 -0.205786747 1.0477479  
## [3,] 1.78978457 -0.48256217 -1.44015162 -0.621785762 1.2529998  
## [4,] 1.25342831 -0.62346170 -0.56272591 -0.413786254 -0.1768099  
## [5,] 0.54095506 -0.31524397 0.06400674 0.418211776 -0.8628212  
## [6,] 0.47691252 -0.67629902 -0.93876550 -0.413786254 1.1851420  
## [7,] 1.09332196 0.86038651 -0.06133979 -0.413786254 1.2398119  
## [8,] 1.78177925 0.75471186 -0.31203285 -0.309786500 1.3431572  
## [9,] 0.98124752 0.11626085 -0.56272591 -0.413786254 1.3553860  
## [10,] 1.02127410 -0.24039109 -0.06133979 0.210212268 1.2045642  
## [11,] 1.35749743 0.60940922 -0.31203285 -0.517786008 1.3779253  
## [12,] -0.48372556 0.49933146 0.56539286 0.106212514 -0.7822551  
## [13,] -0.25957667 0.74150253 -0.31203285 -0.413786254 -0.9321177  
## [14,] -0.09947033 0.44209102 0.31469980 0.106212514 -0.8803252  
## [15,] -0.67585318 0.82075852 0.69073939 -0.621785762 -0.9189298  
## [16,] -0.14750223 0.96606116 0.18935327 -0.933785023 -0.4062796  
## [17,] 0.12467856 -0.73353946 0.06400674 -0.309786500 -0.9074203  
## [18,] -1.15617222 -1.37639358 1.19212551 1.770208574 -0.6961739  
## [19,] -1.26824666 -1.56132422 1.81885816 0.938210544 -0.9349951  
## [20,] -0.40367239 0.98367360 0.31469980 -0.205786747 -0.9184502  
## [21,] -0.37165112 0.40246303 0.69073939 -0.101786993 -0.8611428  
## [22,] -0.41167770 0.19551684 -0.31203285 -0.829785269 -0.8781672  
## [23,] -1.08412436 0.01058620 0.69073939 0.106212514 -0.6235205  
## [24,] -1.02008182 0.15148573 0.44004633 -0.517786008 -0.8858401  
## [25,] 0.18872110 0.90441762 -0.56272591 -1.037784776 -0.9290006  
## [26,] 0.32481149 0.32761015 -0.31203285 -0.309786500 -0.8611428  
## [27,] 0.14869451 0.56537811 -1.31480509 -0.829785269 -0.9251641  
## [28,] -0.77191698 0.44649413 -0.06133979 -0.101786993 -0.8014375  
## [29,] -0.94803397 -1.05056341 1.06677898 0.834210790 -0.5439135  
## [30,] -0.37965643 -0.05105734 -0.18668632 -0.309786500 -0.8402819  
## [31,] 0.42888062 1.55167485 -1.69084468 -0.933785023 1.4043012  
## [32,] -0.06744906 1.50764374 -1.18945856 -0.517786008 1.4035818  
## [33,] 0.57297633 1.71899304 -1.44015162 -0.933785023 1.4112548  
## [34,] 0.02861475 1.21263535 -0.56272591 -0.517786008 1.3863177  
## [35,] 0.74108800 0.62702166 -1.18945856 -0.829785269 1.3685739  
## [36,] -0.41167770 1.11136381 -0.93876550 -0.205786747 1.3887155  
## [37,] 1.76576862 1.49443441 -1.69084468 -0.829785269 1.3983067  
## [38,] 0.73308268 0.54776567 -0.43737938 0.210212268 1.3352445  
## [39,] 0.02060943 -0.53980260 1.19212551 0.106212514 -0.8517913  
## [40,] -0.35564048 -1.89155750 1.56816510 0.002212761 -0.6558909  
## [41,] 0.20473173 -1.49527756 0.69073939 0.210212268 -0.7175144  
## [42,] -0.45970961 -1.45564957 2.32024428 2.186207589 -0.5808396  
## [43,] 0.59699228 -0.57502748 -0.18668632 -0.413786254 -0.8673770  
## [44,] -0.20353945 -1.40721535 1.06677898 -0.413786254 -0.7791379  
## [45,] -2.81327291 -2.17775968 2.57093734 4.370202417 0.0286818  
## [46,] -2.03675713 -0.17874755 -0.06133979 1.874208328 0.2226640  
## [47,] -2.18885816 -1.01093542 0.69073939 1.874208328 0.4120904  
## [,6]  
## [1,] 0.77503669  
## [2,] 0.77503669  
## [3,] 0.08838778  
## [4,] 0.12272023  
## [5,] 0.22571757  
## [6,] 2.28566429  
## [7,] 1.25569093  
## [8,] 1.70201271  
## [9,] 0.36304735  
## [10,] 1.53035049  
## [11,] 1.56468293  
## [12,] -1.18191269  
## [13,] -0.28926911  
## [14,] 0.94669892  
## [15,] -0.42659890  
## [16,] 0.43171224  
## [17,] 0.01972289  
## [18,] 0.08838778  
## [19,] -3.13886208  
## [20,] 0.01972289  
## [21,] -0.66692601  
## [22,] 0.84370158  
## [23,] -1.11324780  
## [24,] -1.59390204  
## [25,] 0.36304735  
## [26,] 1.32435582  
## [27,] -0.66692601  
## [28,] -1.25057758  
## [29,] 0.32871490  
## [30,] 0.87803403  
## [31,] -1.66256693  
## [32,] -0.04894200  
## [33,] -0.56392868  
## [34,] -0.18627178  
## [35,] 0.08838778  
## [36,] -0.73559090  
## [37,] -1.25057758  
## [38,] -0.63259357  
## [39,] 0.12272023  
## [40,] 0.19138512  
## [41,] -0.35793400  
## [42,] 1.04969625  
## [43,] 0.01972289  
## [44,] -0.15193933  
## [45,] -0.66692601  
## [46,] -0.59826112  
## [47,] -0.22060422

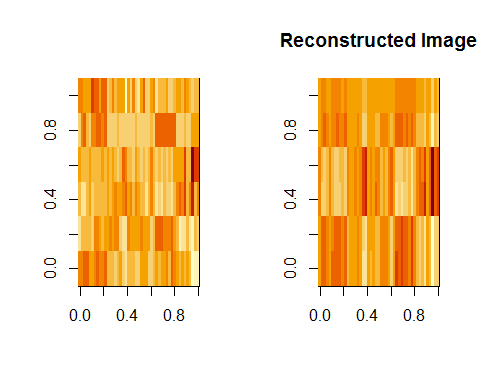
variance.explained = prop.table(swiss.svd$d^2) # varianza spiegata dalle colonne delle matrici U (righe mat. originaria) e V (colonne mat. originaria)  
cumsum(variance.explained) # varianza cumulata spiegata. la prima da sola spiega l'53% con la seconda si arriva al 73%

## [1] 0.5332928 0.7313442 0.8726125 0.9457673 0.9798562 1.0000000

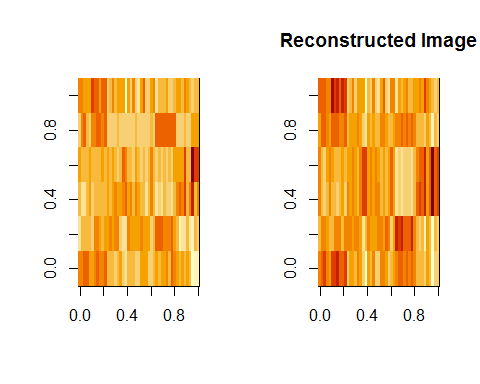
plot(cumsum(swiss.svd$d^2/sum(swiss.svd$d^2)), type="l", xlab="Singular Vector", ylab = "Varianza Cumulata spiegata")



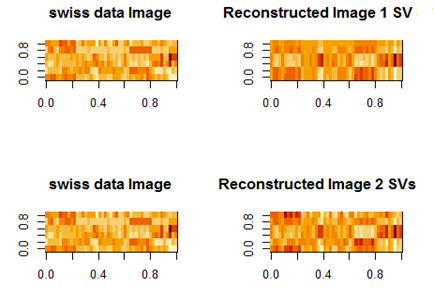
# ora ricostruiamo i dati con un solo Singular Vector (che ci da l'53% della spiegazione totale)  
swiss.recon\_53 <- swiss.svd$u[,1] %\*% diag(swiss.svd$d[1], length(1), length(1)) %\*% t(swiss.svd$v[,1])  
# comparazione grafica  
par(mfrow=c(1,2))  
image(as.matrix((swiss), main="swiss data Image"))  
image(swiss.recon\_53, main="Reconstructed Image")



# ora ricostruiamo i dati con 2 Singular Vector (che ci da il 73% della spiegazione totale)  
swiss.recon\_73 <- swiss.svd$u[,1:2] %\*% diag(swiss.svd$d[1:2]) %\*% t(swiss.svd$v[,1:2])  
# comparazione grafica  
par(mfrow=c(1,2))  
image(as.matrix((swiss), main="swiss data Image"))  
image(swiss.recon\_73, main="Reconstructed Image")



# confrontro tra i dati originari e 1 e poi 2 Singular Vectors.   
# Si vede che con due aumenta la somiglianza con la matrice origianria   
par(mfrow=c(2,2))  
image(as.matrix(swiss), main="swiss data Image")  
image(swiss.recon\_53, main="Reconstructed Image 1 SV 53%")  
image(as.matrix(swiss), main="swiss data Image")  
image(swiss.recon\_73, main="Reconstructed Image 2 SVs 73%")



# Calcolo dell' RMSE  
p <- swiss.svd$u %\*% t(swiss.svd$v)  
dim(p)

## [1] 47 6

RMSE\_UV <- sqrt(mean((data.matrix(swiss) - p)^2)) # siccome ci sono tutti gli elemti della amtrice, va bene mean  
RMSE\_UV

## [1] 0.8642286

# Di seguito un breve esempio per testare il corretto calcolo del RMSE  
#M <- c(5, 2, 4, 4, 3, 3, 1, 2, 4, 1, 2, NA,3, 1, 4, 2, 5, 4, 3, 5, 4, 4, 5, 4, NA)  
M <- c(5, 2, 4, 4, 3, 3, 1, 2, 4, 1, 2, 0, 3, 1, 4, 2, 5, 4, 3, 5, 4, 4, 5, 4, 0)  
M <- t(matrix(M,5,5))  
M

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 5 2 4 4 3  
## [2,] 3 1 2 4 1  
## [3,] 2 0 3 1 4  
## [4,] 2 5 4 3 5  
## [5,] 4 4 5 4 0

M1 <- c(2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0)  
M1 <- t(matrix(M1,5,5))  
M1

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 2 2 2 2 2  
## [2,] 2 2 2 2 2  
## [3,] 2 0 2 2 2  
## [4,] 2 2 2 2 2  
## [5,] 2 2 2 2 0

RMSE <- sqrt( mean((M - M1)^2) ) # corretta se ci sono tutti gli elementi nella matrice  
RMSE2 <- sqrt( sum((M - M1)^2)/23 ) # ok giusto occorre che l'elemto NA abbia un equivalente 0 o NA nelle matrici U e V  
RMSE

## [1] 1.732051

RMSE2

## [1] 1.805788

# per studiare la correlazione del primo e il secondo singular Vector ricostruiamo la matrice originaria solo utilizzando la prima riga U e la prima colonna V  
# 53% è la spiegazione in caso di standardizzazione  
  
swiss.recon\_53 <- swiss.svd$u[,1] %\*% diag(swiss.svd$d[1], length(1), length(1)) %\*% t(swiss.svd$v[,1])  
#swiss.recon\_53  
cor(swiss,swiss.recon\_53)[,1] # Examination -.91

## Fertility Agriculture Examination Education   
## 0.8174532 0.7588284 -0.9118030 -0.8126670   
## Catholic Infant.Mortality   
## 0.6262741 0.2677219

swiss.recon\_73 <- swiss.svd$u[,2] %\*% diag(swiss.svd$d[2], length(1), length(1)) %\*% t(swiss.svd$v[,2])  
#swiss.recon\_73  
cor(swiss,swiss.recon\_73)[,2] # Infant.Mortality -.88

## Fertility Agriculture Examination Education   
## -0.3510417 0.4485887 -0.1362802 -0.1951810   
## Catholic Infant.Mortality   
## -0.1590155 -0.8842468

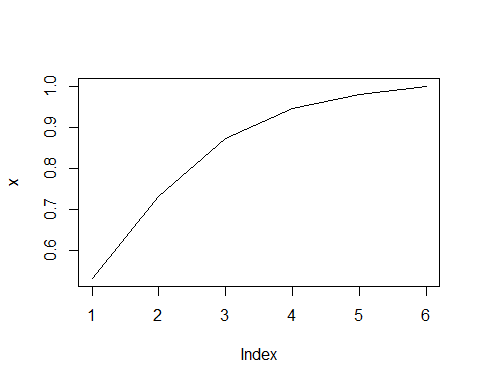
swiss.recon\_87 <- swiss.svd$u[,3] %\*% diag(swiss.svd$d[3], length(1), length(1)) %\*% t(swiss.svd$v[,3])  
#swiss.recon\_87  
cor(swiss,swiss.recon\_87)[,3] # Catholic -.74

## Fertility Agriculture Examination Education   
## 0.15997928 -0.03530235 0.08399797 -0.49015164   
## Catholic Infant.Mortality   
## -0.74279084 0.14740308

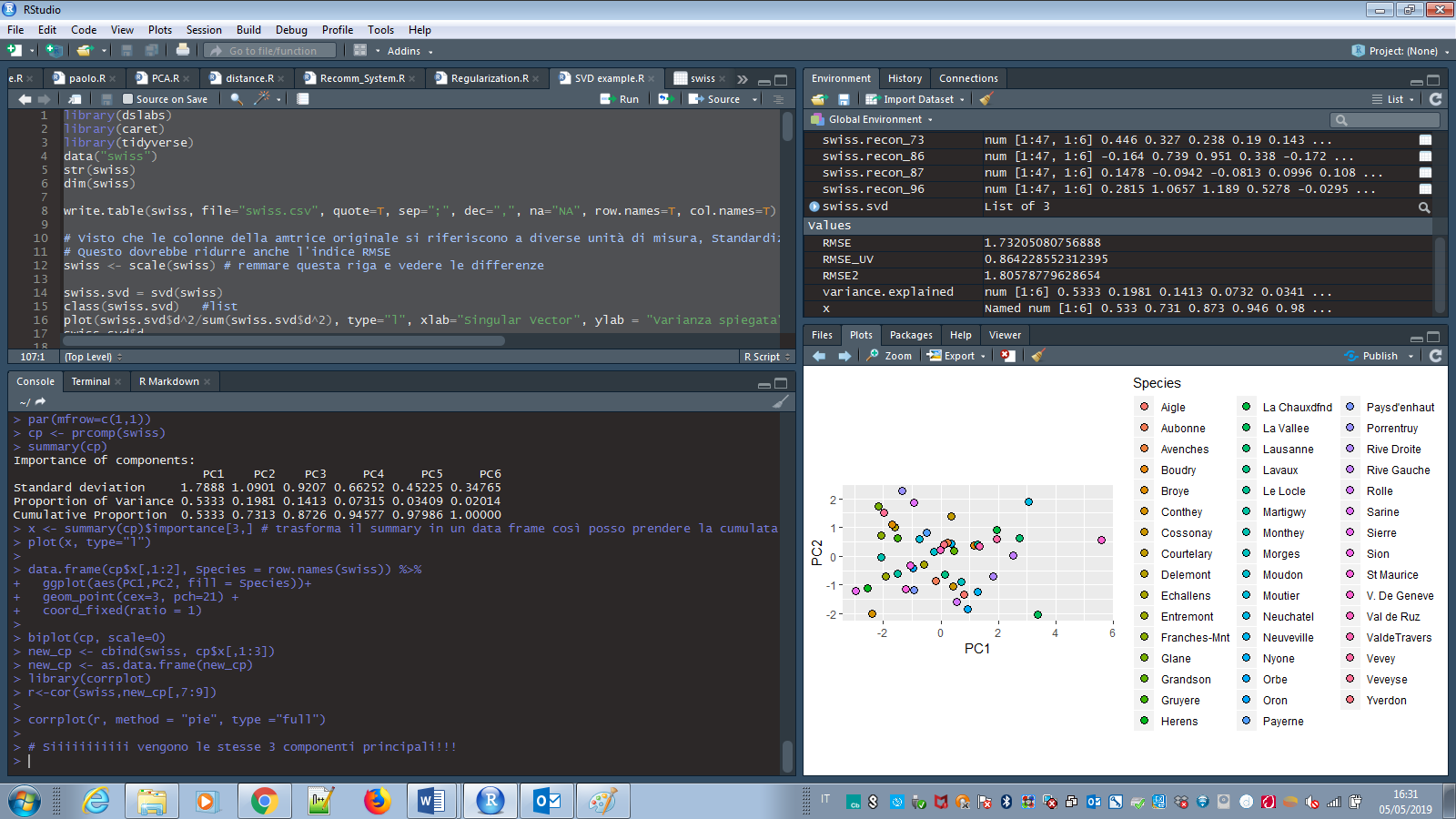
# Ora verificare se queste 3 singular vectors corrispondono alla stesse se invece si utilizza PCA  
  
par(mfrow=c(1,1))  
cp <- prcomp(swiss)  
summary(cp)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 1.7888 1.0901 0.9207 0.66252 0.45225 0.34765  
## Proportion of Variance 0.5333 0.1981 0.1413 0.07315 0.03409 0.02014  
## Cumulative Proportion 0.5333 0.7313 0.8726 0.94577 0.97986 1.00000

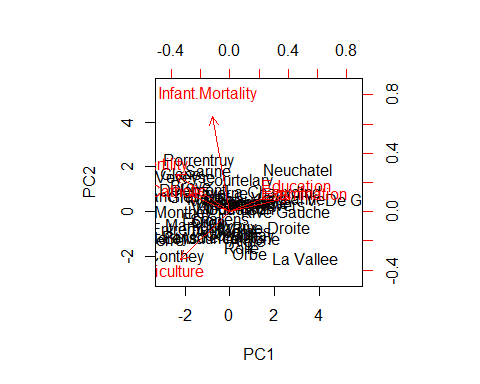
x <- summary(cp)$importance[3,] # trasforma il summary in un data frame così posso prendere la cumulata  
plot(x, type="l")



data.frame(cp$x[,1:2], Species = row.names(swiss)) %>%   
 ggplot(aes(PC1,PC2, fill = Species))+  
 geom\_point(cex=3, pch=21) +  
 coord\_fixed(ratio = 1)



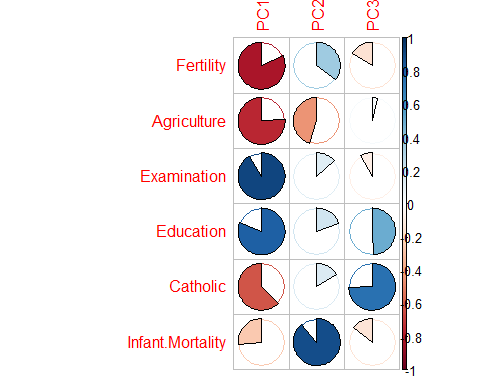
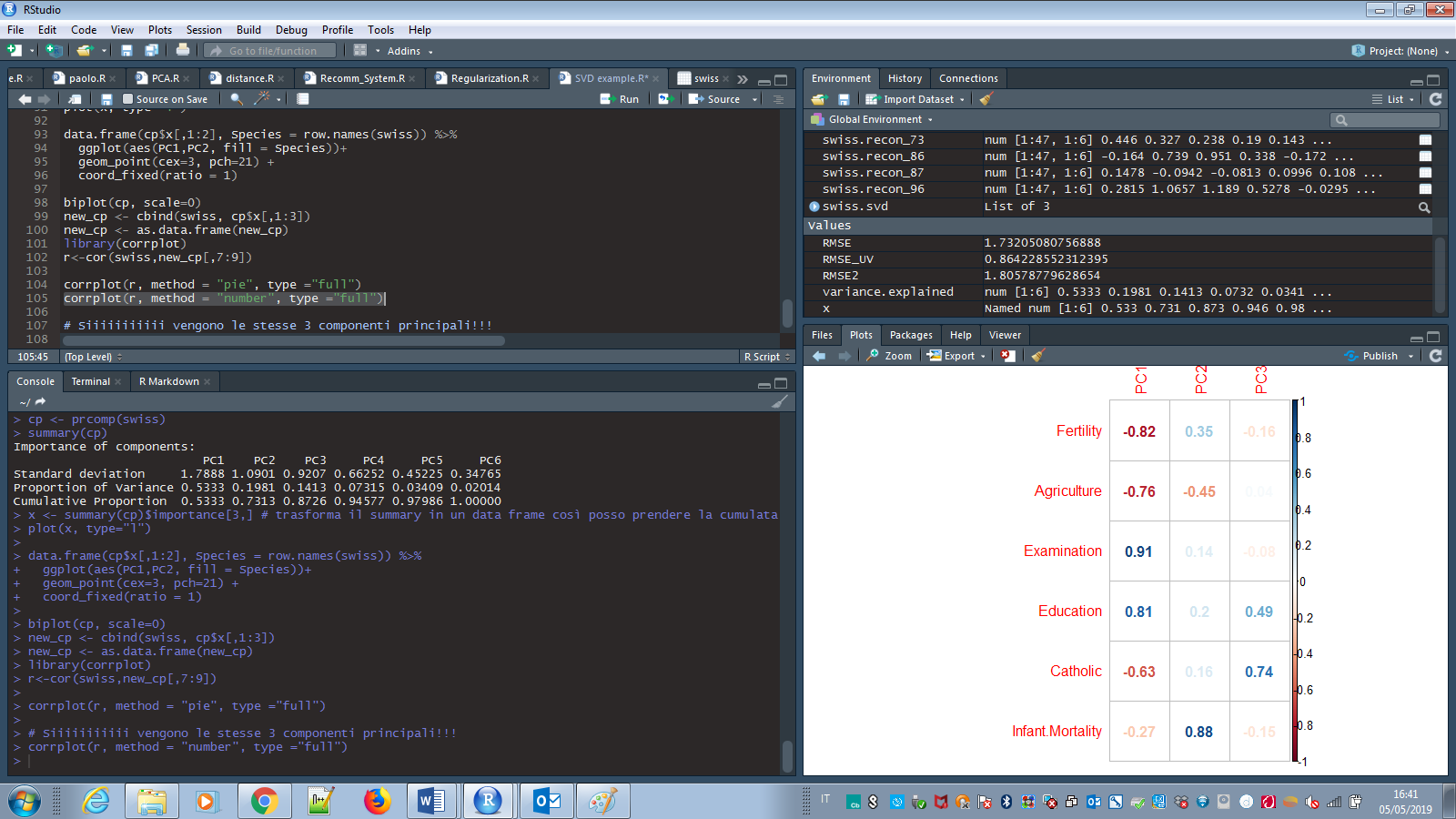
biplot(cp, scale=0)



new\_cp <- cbind(swiss, cp$x[,1:3])  
new\_cp <- as.data.frame(new\_cp)  
library(corrplot)

## corrplot 0.84 loaded

r<-cor(swiss,new\_cp[,7:9])  
  
corrplot(r, method = "pie", type ="full") # try “number” or “pie”

# Siiiiiiiiiii vengono le stesse 3 componenti principali!!!

PC1=Examination (0.91);

PC2=Infant. Mortality (0.88);

PC3=Catholic (0.74)

# per dettagli sul corrplot si veda:

<https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html>