Introdução a análise de componentes principais

Guilherme Ludwig

September 11, 2019

Motivação

Prós

- Reduzir a dimensão dos dados.
- Muitas vezes, componentes principais servem de input para regressão (PCR, PLS na aula que vem) e clustering (em breve).
- Pode servir de remédio para multicolinearidade (mas com cuidado) e seleção de variáveis.

Contras

- Alguns componentes principais são de difícil interpretação.

Componentes Principais

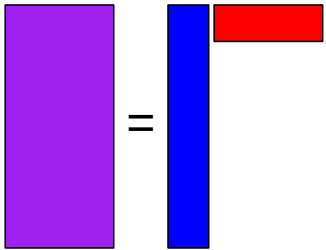
Seja $\hat{\mathbf{\Sigma}} = \mathbf{S}$, com decomposição espectral $\mathbf{S} = \hat{\mathbf{Q}}\hat{\mathbf{\Lambda}}\hat{\mathbf{Q}}^t$. Seja $\mathbf{A} = (n-1)^{-1/2}(\mathbf{X} - \mathbf{1}_{n\times 1}\bar{\mathbf{X}}_{1\times p})$ com decomposição SVD $\mathbf{A} = \mathbf{UDV}^t$.

Naturalmente, $\mathbf{A}^t\mathbf{A} = \mathbf{S}$, logo $\mathbf{VD}^2\mathbf{V}^t = \hat{\mathbf{Q}}\hat{\mathbf{\Lambda}}\hat{\mathbf{Q}}^t$.

Para deixar claro (a menos de algumas contantes de escala):

$$\mathbf{Z} = \underbrace{\mathbf{UD}}_{\text{Scores}} \quad \underbrace{\mathbf{V}^t}_{\text{Loadings}}$$

Esparsidade



Exemplo: Vidro

> *The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence... if it is correctly identified!*

O conjunto de dados pode ser obtido em https://www.kaggle.com/uciml/glass.

Exemplo: Vidro

```
RI: refractive index
Na: Sodium (unit measurement: weight percent in corresponding oxide, as ar
Mg: Magnesium
Al: Aluminum
Si: Silicon
K: Potassium
Ca: Calcium
Ba: Barium
Fe: Iron
Type of glass: -- 1 building_windows_float_processed
               -- 2 building_windows_non_float_processed
               -- 3 vehicle_windows_float_processed
               -- 4 vehicle_windows_non_float_processed (none in this datal
               -- 5 containers
               -- 6 tableware
               -- 7 headlamps
```

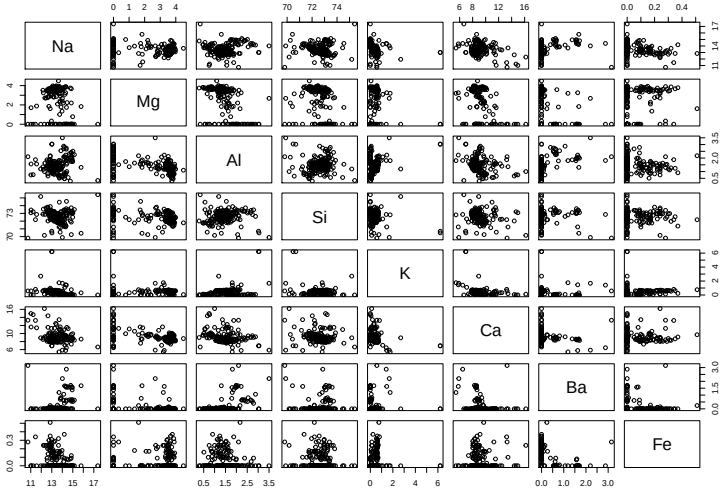
Exemplo: Vidro

Eu vou remover `RI` e `Type`, mas usaremos novamente esses dados para classificação!

```
str(glass <- read.csv("data/glass.csv"))
```

```
## 'data.frame':    214 obs. of  10 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
##  $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: int  1 1 1 1 1 1 1 1 1 1 ...
```

Exemplo: Vidro



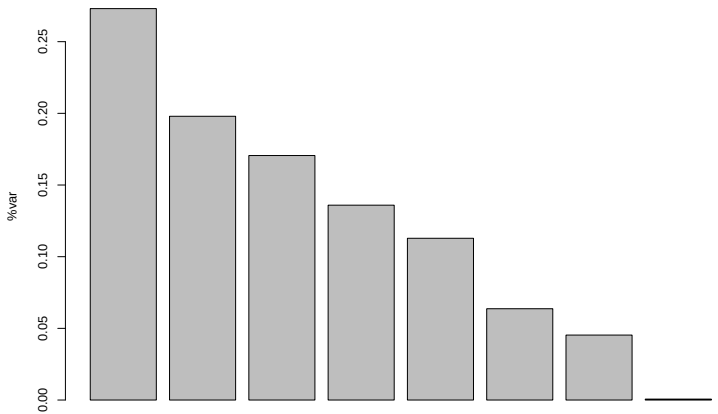
Exemplo: Vidro

```
glass0 <- glass[,-c(1,10)]
glass.scaled <- scale(glass0,
                      center = TRUE, scale = TRUE)
(Rho <- round(var(glass.scaled),2))
```

```
##        Na    Mg    Al    Si     K    Ca    Ba    Fe
## Na   1.00 -0.27  0.16 -0.07 -0.27 -0.28  0.33 -0.24
## Mg  -0.27  1.00 -0.48 -0.17  0.01 -0.44 -0.49  0.08
## Al   0.16 -0.48  1.00 -0.01  0.33 -0.26  0.48 -0.07
## Si  -0.07 -0.17 -0.01  1.00 -0.19 -0.21 -0.10 -0.09
## K   -0.27  0.01  0.33 -0.19  1.00 -0.32 -0.04 -0.01
## Ca  -0.28 -0.44 -0.26 -0.21 -0.32  1.00 -0.11  0.12
## Ba   0.33 -0.49  0.48 -0.10 -0.04 -0.11  1.00 -0.06
## Fe  -0.24  0.08 -0.07 -0.09 -0.01  0.12 -0.06  1.00
```

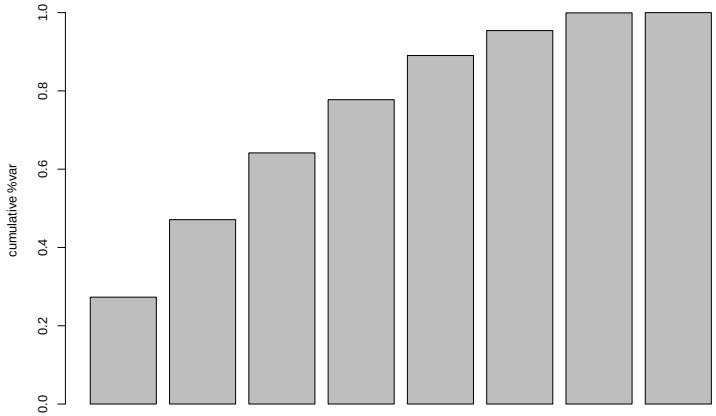
Exemplo: Vidro

```
pc <- eigen(Rho)
barplot(pc$values/sum(pc$values), ylab = "%var")
```



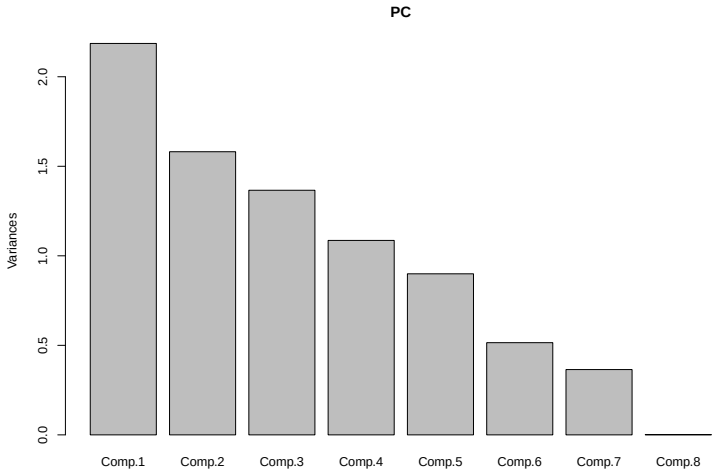
Exemplo: Vidro

```
barplot(cumsum(pc$values)/sum(pc$values),
        ylab = "cumulative %var")
```



Exemplo: Vidro

```
PC <- princomp(glass0, cor = TRUE)
plot(PC)
```

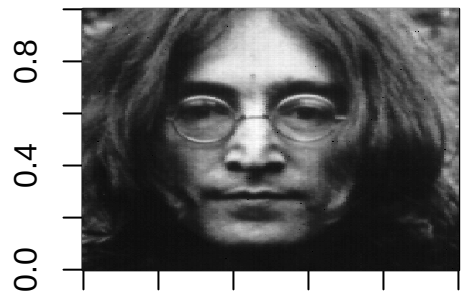


loadings(PC)

```
##
## Loadings:
##    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## Na -0.378         0.477  0.418  0.104  0.575  0.112 -0.316
## Mg  0.497  0.402  0.192  0.267  0.122 -0.302 -0.202 -0.580
## Al -0.521  0.230 -0.264 -0.164                -0.731 -0.201
## Si                0.391 -0.829  0.193         0.157 -0.310
## K          0.539 -0.489 -0.105 -0.277  0.285  0.478 -0.264
## Ca  0.107 -0.692 -0.323        -0.285                -0.566
## Ba -0.536                0.167  0.244 -0.652  0.393 -0.194
## Fe  0.185 -0.104 -0.407         0.847  0.262
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.125  0.125  0.125  0.125  0.125  0.125  0.125  0.125
## Cumulative Var  0.125  0.250  0.375  0.500  0.625  0.750  0.875  1.000
```

Reconstrução de imagens

```
library(fields)
data(lennon)
image(lennon, col = grey(seq(0, 1, length.out = 256)))
```



Reconstrução de imagens

```
dim(lennon)
```

```
## [1] 256 256
```

```
lennon[1:5,1:5]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   14   14   15   16   17
## [2,]   14   13   14   14   14
## [3,]   12   14   13   16   16
## [4,]   14   14   14   15   16
## [5,]   13   13   13   13   14
```

Eigen-Lennon:

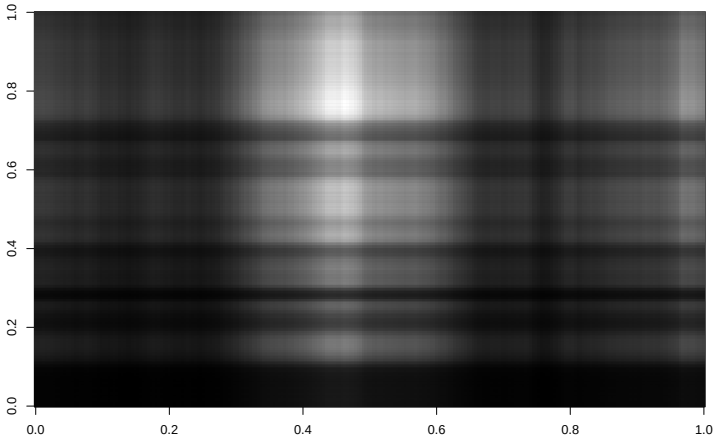
Estou fazendo $\texttt{lennon}^t\texttt{lennon}$

```
eigLen <- svd(lennon)
```

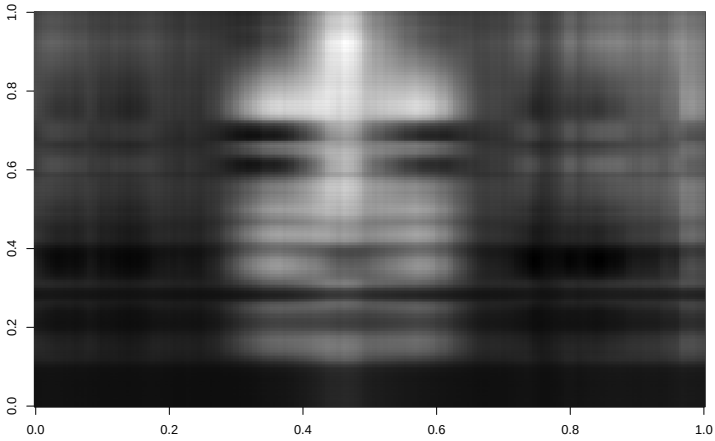
Para cada j, vou fazer

```
image <- matrix(0, 256, 256)
for(j in 1:q){
  image <- image + eigLen$d[j]*outer(eigLen$u[,j],
                                     eigLen$v[,j])
}
```

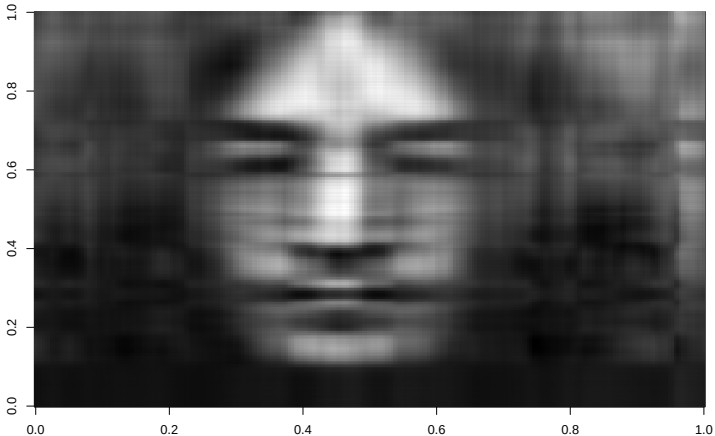
$q = 1$



$q = 2$



$q = 4$



$q = 8$

$q = 16$

Note

```
object.size(lennon)
```

```
## 524488 bytes
```

```
object.size(eigLen$d[1:16]) +
  object.size(eigLen$u[,1:16]) +
  object.size(eigLen$u[,1:16])
```

```
## 66104 bytes
```