R Notebook

Code

Nome: Vinícius de Oliveira Silva

Matrícula: 2013007820

Questão 1

a)

Provar que $\max_{x \neq 0} \frac{x'Bx}{x'x} = \lambda_1$ é obtido quando $x = v_1$

↳ Prova:

- Seja P a matriz composta pelos autovetores de B:

  $P = [v_1, v_2, \dots v_p]$

- Seja A uma matriz diagonal onde seus elementos diagonais são os autovalores de B. $A = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & \cdots \\ 0 & \lambda_2 & 0 & 0 & \cdots \\ 0 & 0 & \lambda_3 & 0 & \cdots \\ 0 & 0 & 0 & \ddots & \\ \vdots & \vdots & \vdots & & \lambda_p \end{bmatrix}$

- Seja $B^{\frac{1}{2}}$ a matriz tal que $B^{\frac{1}{2}} B^{\frac{1}{2}} = B$. $\Rightarrow$ Pelo teorema espectral, $B^{\frac{1}{2}} = PA^{\frac{1}{2}}P'$

- Seja $Y = P'X \Rightarrow$ Se $x \neq 0$, $Y \neq 0$

Temos então:

$$\frac{x'Bx}{x'x} = \frac{x' B^{\frac{1}{2}} B^{\frac{1}{2}} X}{x' \underbrace{PP'}_{\text{Matriz } I} X} = \frac{\overbrace{x'P}^{Y'} A^{\frac{1}{2}} \overbrace{P'P}^{I} A^{\frac{1}{2}} \overbrace{P'X}^{Y'}}{Y'Y} = \frac{Y'AY}{Y'Y} =$$

$$\sum_{i=1}^{p} \frac{\lambda_i Y_i^2}{\sum_{i=1}^{p} Y_i^2} \leq \lambda_1 \frac{\sum_{i=1}^{p} Y_i^2}{\sum_{i=1}^{p} Y_i^2} = \lambda_1$$

Provamos que a razão $\frac{x'Bx}{x'x}$ é no máximo igual a λ_1

Precisamos ainda provar que a razão atinge esse máximo quando $x = v_1$:

Assumindo que $X = v_1$, temos:

$Y = P'v_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ → Sabemos que o produto interno entre 2 autovetores distintos é zero pois eles são ortogonais

Assim, o produto $Y'Y = 1$, o que nos permite escrever:

$$\frac{Y'AY}{Y'Y} = \frac{Y'AY}{1} = \lambda_1$$

Tendo mostrado que λ_1 é o maior valor que $\frac{x'BX}{x'x}$ pode atingir, que $\frac{x'Bx}{x'x} = \frac{Y'AY}{Y'Y}$ e que $\frac{Y'AY}{Y'Y} = \lambda_1$ quando $X = v_1$, podemos concluir que a razão é máxima (λ_1) quando $x = v_1$.

b)

Seja $Y = l_1 X_1 + l_2 X_2 + \dots l_p X_p$

↳ Y é uma variável aleatória de variância $V(Y)$.

Queremos provar que $V(Y)$ é máxima quando os coeficientes $[l_1, l_2, \dots l_p]$ formam o autovetor v_1 da matriz de covariância de X. Neste caso, $V(Y) = \lambda_1$

Prova:

Utilizando a definição de variância, temos que:

$$\begin{aligned}
V(Y) &= E(Y - E(Y))^2 \\
&= E((l_1X_1 + l_2X_2 + \dots + l_kX_k) - (l_1\mu_1 + l_2\mu_2 + \dots l_k\mu_k))^2 \\
&= E(l_1(X_1 - \mu_1) + l_2(X_2 - \mu_2) + \dots + l_k(X_k - \mu_k))^2 \\
&= E\Big(\sum_i l_i^2 (X_i - \mu_i)^2 + \sum_{i \neq j} l_i l_j (X_i - \mu_i)(X_j - \mu_j)\Big) \\
&= \sum_i E(l_i^2 (X_i - \mu_i)^2) + \sum_{i \neq j} E(l_i l_j (X_i - \mu_i)(X_j - \mu_j)) \\
&= \sum_i l_i^2 E(X_i - \mu_i)^2 + \sum_{i \neq j} l_i l_j E((X_i - \mu_i)(X_j - \mu_j)) \\
&= \sum_i l_i^2 V(X_i) + \sum l_i l_j \, Cov(X_i, X_j)
\end{aligned}$$

Abrindo o quadrado

Linearidade da esperança

Linearidade da esperança

Definição de Variância e Covariância

Assim, temos:

$$V(Y) = (l_1, l_2, \dots l_k) \cdot \Sigma \cdot \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_k \end{pmatrix}$$

$$V(Y) = l' \Sigma l$$

Considerando que $l'l \neq 0$, podemos definir a razão:

$$\frac{l' \Sigma l}{l'l}$$

No item a) já provamos que essa razão é máxima quando $l = v_1$ e seu valor é λ_1.

Questão 2 - Exercício 8.11 - Johnson Wichern

a)

Hide

```
#leitura da tabela original
table = read.table("T8-5.DAT", header=FALSE)
#extraindo a matriz de covariancia original
estimatedCovTransform <- cov(table)
#multiplicando a ultima coluna por 10
transformedTable <- table
transformedTable[,5] <- transformedTable[,5]*10
#extraindo a nova matriz de covariancia
covTransform <- cov(transformedTable)
#obtendo a nova matriz de cov a partir da original
estimatedCovTransform[,5] <- estimatedCovTransform[,5]*10
estimatedCovTransform[5,] <- estimatedCovTransform[5,]*10
#Checando nossa aproximacao:
expectedZeros <- round(estimatedCovTransform - covTransform)
sprintf("Essa matriz deveria ser composta apenas por zeros:")
```

```
[1] "Essa matriz deveria ser composta apenas por zeros:"
```

Hide

```
prmatrix(expectedZeros, rowlab=rep("",5), collab=rep("",5))
```

```

 0 0 0 0 0
 0 0 0 0 0
 0 0 0 0 0
 0 0 0 0 0
 0 0 0 0 0
```

O procedimento de multiplicar a quinta coluna e a quinta coluna funciona porque a variancia da quinta variável segue a fórmula:

$$V(c'X) = c'V(X)c$$

como c é uma constante numérica igual a 10, a nova variância fica multiplicada por c^2 = 100. Os elementos da quinta coluna e da quinta linha dependem da raíz quadrada da nova variância, que é 10 * (variância original)

b)

Hide

```
#calculo do PCA propriamente dito
pca <- prcomp(t(transformedTable))
pca
```

```
Standard deviations (1, .., p=5):
[1] 2.186664e+02 3.580045e+01 1.768318e+01 8.411473e+00 2.190192e-14

Rotation (n x k) = (61 x 5):
             PC1          PC2           PC3           PC4          PC5
 [1,] 0.12337779 -0.065572228 -0.0046970376  0.1447320391 -0.736323336
 [2,] 0.13358423 -0.218068539 -0.0523294924  0.0846350281 -0.218417830
 [3,] 0.10924984 -0.075368354  0.1353766934  0.2060795049  0.400329247
 [4,] 0.12258874  0.036996943  0.0468777912  0.1082723393 -0.016495114
 [5,] 0.12695274 -0.016082657  0.0969188416  0.1099338678  0.077630909
 [6,] 0.09332611 -0.333943225 -0.0137214800 -0.1091080640  0.016588163
 [7,] 0.12024213 -0.162901535 -0.0150973504  0.0520901317  0.036504501
 [8,] 0.12360291 -0.163663278 -0.0269393794 -0.0151849664  0.056874687
 [9,] 0.14659846  0.137612380  0.0733516045 -0.0220376561 -0.006171418
[10,] 0.12876943  0.030396928 -0.0855174877 -0.0952806771  0.025372870
[11,] 0.11335203 -0.045003182 -0.0320975839  0.0227926654  0.017094681
[12,] 0.14847395  0.113501495 -0.0578129793  0.1032383460 -0.001582688
[13,] 0.15165369  0.115477390 -0.0758707774  0.1068367931  0.033075415
[14,] 0.12858268  0.094970082 -0.0475079230  0.0002000893 -0.017627894
[15,] 0.12262223 -0.028098612  0.0977483924  0.0606840839 -0.097668685
[16,] 0.12283867 -0.132063599 -0.1146968069  0.0077365327  0.006877333
[17,] 0.12026266 -0.182722371  0.0858761756  0.2639913584 -0.001992861
[18,] 0.14062606 -0.042992827 -0.0988828936 -0.0123083794  0.055719250
[19,] 0.11712182 -0.158957762 -0.0147661112 -0.1806156178  0.067672418
[20,] 0.12455521 -0.126707820 -0.0701113559  0.0048898163  0.073049176
[21,] 0.11322910  0.056036342 -0.1346005546 -0.0567397328  0.018613619
[22,] 0.12862601  0.083215320  0.0124690343 -0.0351486915 -0.009271533
[23,] 0.12538379  0.181268378  0.4732852706 -0.2833284497 -0.033249830
[24,] 0.12126280  0.085237238  0.0467240149  0.1496343007 -0.109822658
[25,] 0.14091621  0.070861459 -0.0843635184 -0.0559028018 -0.016362256
[26,] 0.09349727 -0.058564403  0.3406348416 -0.4008343516 -0.101542612
[27,] 0.11272977 -0.033156868  0.0319659202 -0.1818108723 -0.064120194
[28,] 0.14144293 -0.070637312 -0.1428728757  0.0040079120  0.141623693
[29,] 0.15066253 -0.015283991 -0.0917973479  0.0292068506 -0.029614870
[30,] 0.13737132  0.025164651 -0.1548956886  0.0396426906  0.039365722
[31,] 0.13682198  0.021916576 -0.1569136541  0.0258266086 -0.050098620
[32,] 0.11939926 -0.008426196 -0.0924197821 -0.0198210525  0.070377716
[33,] 0.12496482 -0.031904829 -0.0593426161  0.0178194866  0.129990210
[34,] 0.08915896 -0.045073685  0.1291348000 -0.1377317196 -0.011917472
[35,] 0.13889753 -0.013974186 -0.0996594034  0.0031684396  0.047542615
[36,] 0.12145452  0.011317834 -0.0441961326 -0.0540230631 -0.131030065
[37,] 0.11164348 -0.016976311 -0.0061341530 -0.0106335057 -0.142868425
[38,] 0.13080181 -0.012523253 -0.0836848732  0.0838598437  0.040847424
[39,] 0.14131888  0.129430129 -0.0808001445 -0.0258881718  0.073659571
[40,] 0.15057861  0.152583069 -0.0121424306  0.0138992394 -0.018364788
[41,] 0.12434321 -0.057557527 -0.1015607808 -0.0060071369 -0.018088359
[42,] 0.13029300  0.010536083 -0.0598088083  0.1100623044 -0.077982281
[43,] 0.13109090  0.066618975 -0.0493981271  0.0074321964 -0.042546247
[44,] 0.13334220 -0.053600459 -0.0835112264 -0.1048909047  0.081791172
[45,] 0.13700882  0.110877346 -0.0460366048  0.0131795171  0.019605550
[46,] 0.14156382  0.067930207 -0.1083505731 -0.0862210628  0.077232504
[47,] 0.10410673 -0.563871229  0.1007026924 -0.2177664181  0.047128665
[48,] 0.10217042 -0.261297468  0.3403301113  0.3167289737  0.053926169
[49,] 0.10711562  0.064809346  0.3376535347  0.3521117180  0.098256509
[50,] 0.11821614 -0.031683765 -0.0336978131  0.0342316920 -0.071745039
[51,] 0.11972116  0.012588222  0.0083215468 -0.0291725154 -0.119073548
[52,] 0.14125506  0.060785439 -0.0799233083 -0.1522696511  0.154184788
```

```
[53,] 0.13111357  0.058718779  0.0001352152 -0.0603030591  0.024207081
[54,] 0.13525089  0.040232126  0.0870500850 -0.0878609409  0.062690604
[55,] 0.14494879  0.097036457 -0.0013213482  0.0062808278  0.040823237
[56,] 0.13014925  0.084339319 -0.0452681687 -0.2092890949 -0.030453653
[57,] 0.15166024  0.185774872  0.2854085606  0.0574794280  0.024370142
[58,] 0.13621352  0.130306026  0.1247323002 -0.1048862414  0.014268212
[59,] 0.12562285 -0.005956974  0.0192892162  0.0823988329 -0.042752682
[60,] 0.13895130  0.095326223 -0.0225275313  0.0889642719  0.046516614
[61,] 0.12857748  0.094600908  0.0585270517 -0.0522489065 -0.011761002
```

Duas primeiras componentes principais:

Hide

```
#salvamos os autovalores e os autovetores
eigenvalues <- (pca$sdev)^2
eigenvectors <- (pca$rot)
print(eigenvectors[,c(1,2)])
```

```
             PC1          PC2
 [1,] 0.12337779 -0.065572228
 [2,] 0.13358423 -0.218068539
 [3,] 0.10924984 -0.075368354
 [4,] 0.12258874  0.036996943
 [5,] 0.12695274 -0.016082657
 [6,] 0.09332611 -0.333943225
 [7,] 0.12024213 -0.162901535
 [8,] 0.12360291 -0.163663278
 [9,] 0.14659846  0.137612380
[10,] 0.12876943  0.030396928
[11,] 0.11335203 -0.045003182
[12,] 0.14847395  0.113501495
[13,] 0.15165369  0.115477390
[14,] 0.12858268  0.094970082
[15,] 0.12262223 -0.028098612
[16,] 0.12283867 -0.132063599
[17,] 0.12026266 -0.182722371
[18,] 0.14062606 -0.042992827
[19,] 0.11712182 -0.158957762
[20,] 0.12455521 -0.126707820
[21,] 0.11322910  0.056036342
[22,] 0.12862601  0.083215320
[23,] 0.12538379  0.181268378
[24,] 0.12126280  0.085237238
[25,] 0.14091621  0.070861459
[26,] 0.09349727 -0.058564403
[27,] 0.11272977 -0.033156868
[28,] 0.14144293 -0.070637312
[29,] 0.15066253 -0.015283991
[30,] 0.13737132  0.025164651
[31,] 0.13682198  0.021916576
[32,] 0.11939926 -0.008426196
[33,] 0.12496482 -0.031904829
[34,] 0.08915896 -0.045073685
[35,] 0.13889753 -0.013974186
[36,] 0.12145452  0.011317834
[37,] 0.11164348 -0.016976311
[38,] 0.13080181 -0.012523253
[39,] 0.14131888  0.129430129
[40,] 0.15057861  0.152583069
[41,] 0.12434321 -0.057557527
[42,] 0.13029300  0.010536083
[43,] 0.13109090  0.066618975
[44,] 0.13334220 -0.053600459
[45,] 0.13700882  0.110877346
[46,] 0.14156382  0.067930207
[47,] 0.10410673 -0.563871229
[48,] 0.10217042 -0.261297468
[49,] 0.10711562  0.064809346
[50,] 0.11821614 -0.031683765
[51,] 0.11972116  0.012588222
[52,] 0.14125506  0.060785439
[53,] 0.13111357  0.058718779
[54,] 0.13525089  0.040232126
[55,] 0.14494879  0.097036457
[56,] 0.13014925  0.084339319
```

```
[57,] 0.15166024  0.185774872
[58,] 0.13621352  0.130306026
[59,] 0.12562285 -0.005956974
[60,] 0.13895130  0.095326223
[61,] 0.12857748  0.094600908
```

c)

Hide

```
#proporcao de variancia
varRatio <- sum(eigenvalues[c(1,2)])/sum(eigenvalues)
sprintf("As duas primeiras componentes principais explicam %s%% da variancia", format
(round(varRatio*100, digits = 2), nsmall = 2))
```

```
[1] "As duas primeiras componentes principais explicam 99.23% da variancia"
```

Hide

```
#Computando as correlações:
originalCors <- cor(table)
transformedCors <- cor(transformedTable)
#verificando as diferenças entre as correlações da matriz original e da transformada:
expectedZeros <- round(transformedCors - originalCors)
sprintf("Essa matriz deveria ser composta apenas por zeros:")
```

```
[1] "Essa matriz deveria ser composta apenas por zeros:"
```

Hide

```
prmatrix(expectedZeros, rowlab=rep("",5), collab=rep("",5))
```

```


 0 0 0 0 0
 0 0 0 0 0
 0 0 0 0 0
 0 0 0 0 0
 0 0 0 0 0
```

Percebemos que as correlações não são afetadas quando fazemos uma mudança de escala em uma das variáveis.

Hide

```
#computando os autovetores considerando a matriz original:
#calculo do PCA propriamente dito
pcaOriginal <- prcomp(t(table))
originalEigenvals <- (pcaOriginal$sdev)^2
originalVarRatio <- sum(originalEigenvals[c(1,2)])/sum(originalEigenvals)
#podemos visualizar a proporção de variancia explicada pelos dois primeiros PC's em a
mbos os casos:
sprintf("Na matriz modificada, as duas primeiras componentes principais explicam %s%%
 da variancia", format(round(varRatio*100, digits = 2), nsmall = 2))
```

```
[1] "Na matriz modificada, as duas primeiras componentes principais explicam 99.23% d
a variancia"
```

Hide

```
sprintf("Na matriz original, as duas primeiras componentes principais explicam %s%% d
a variancia", format(round(originalVarRatio*100, digits = 2), nsmall = 2))
```

```
[1] "Na matriz original, as duas primeiras componentes principais explicam 99.79% da
variancia"
```

Como podemos ver, ao alterarmos a escala da última coluna, os dois primeiros componentes passam a explicar ligeiramente menos variância, o que significa que essa coluna passou a ser mais significativa.

Questão 3

Hide

```
sigma <- matrix(c(1.24, 0.48, 0.16, 0.48, 0.86, 0.12, 0.16, 0.12, 0.14), ncol = 3, nr
ow = 3)
l <- c(0.8, 0.6, 0.2)
#Podemos obter a matriz psi através do calculo de Sigma - (LL')
psi <- sigma - (l%*%t(l))
sprintf("A matriz Ψ é:")
```

```
[1] "A matriz Ψ é:"
```

Hide

```
prmatrix(round(psi,2), rowlab=rep("",3), collab=rep("",3))
```

```

 0.6 0.0 0.0
 0.0 0.5 0.0
 0.0 0.0 0.1
```

Questão 4

Hide

```
#URL do arquivo de treinamento
arq = "http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
#abertura do arquivo para leitura
wine=read.table(arq, sep=",")
#visualização dos primeiros registros
head(wine)
```

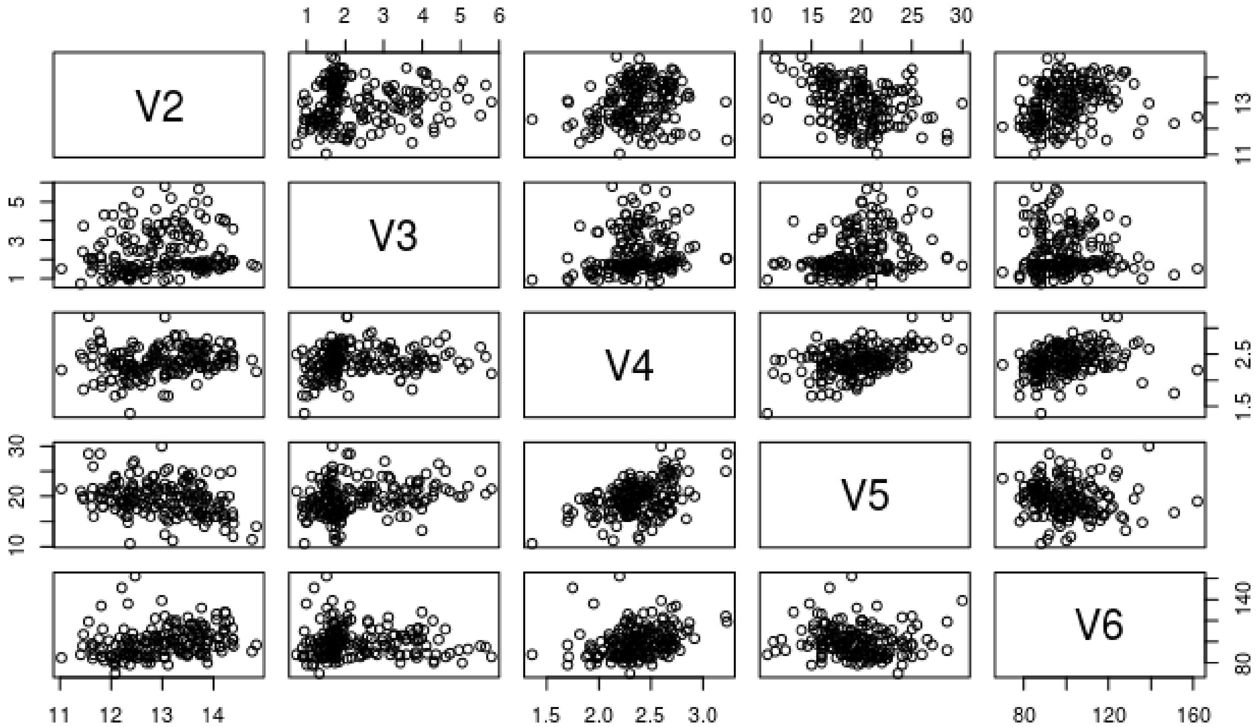
	V1 <int>	V2 <dbl>	V3 <dbl>	V4 <dbl>	V5 <dbl>	V6 <int>	V7 <dbl>	V8 <dbl>	V9 <dbl>
1	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28

	V1 <int>	V2 <dbl>	V3 <dbl>	V4 <dbl>	V5 <dbl>	V6 <int>	V7 <dbl>	V8 <dbl>	V9 <dbl>
2	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26
3	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30
4	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24
5	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39
6	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34

6 rows | 1-10 of 14 columns

Hide

```
#visualização da dispersão entre as variáveis de 2 a 6
pairs(wine[,2:6])
```



Hide

```
#calcula a correlação entre as colunas de 2 a 14 e multiplica por 100
round(100*cor(wine[,2:14]))
```

```
     V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14
V2  100   9  21 -31  27  29  24 -16  14  55  -7   7  64
V3    9 100  16  29  -5 -34 -41  29 -22  25 -56 -37 -19
V4   21  16 100  44  29  13  12  19   1  26  -7   0  22
V5  -31  29  44 100  -8 -32 -35  36 -20   2 -27 -28 -44
V6   27  -5  29  -8 100  21  20 -26  24  20   6   7  39
V7   29 -34  13 -32  21 100  86 -45  61  -6  43  70  50
V8   24 -41  12 -35  20  86 100 -54  65 -17  54  79  49
V9  -16  29  19  36 -26 -45 -54 100 -37  14 -26 -50 -31
V10  14 -22   1 -20  24  61  65 -37 100  -3  30  52  33
V11  55  25  26   2  20  -6 -17  14  -3 100 -52 -43  32
V12  -7 -56  -7 -27   6  43  54 -26  30 -52 100  57  24
V13   7 -37   0 -28   7  70  79 -50  52 -43  57 100  31
V14  64 -19  22 -44  39  50  49 -31  33  32  24  31 100
```

Hide

```
#calcula o desvio padrao de todas as variáveis individualmente
round(apply(wine[,2:14], 2, sd),2)
```

```
    V2      V3      V4      V5      V6      V7      V8      V9     V10     V11     V12     V13     V14
  0.81    1.12    0.27    3.34   14.28    0.63    1.00    0.12    0.57    2.32    0.23    0.71 314.91
```

Hide

```
#calcula os autovetores e os autovalores da matriz de covariancia
wine.pca <- prcomp(wine[,2:14], scale. = TRUE)
#exibe um resumo das componentes principais
summary(wine.pca)
```

```
Importance of components:
                         PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8
  PC9   PC10    PC11    PC12    PC13
Standard deviation     2.169 1.5802 1.2025 0.95863 0.92370 0.80103 0.74231 0.59034 0.
53748 0.5009 0.47517 0.41082 0.32152
Proportion of Variance 0.362 0.1921 0.1112 0.07069 0.06563 0.04936 0.04239 0.02681 0.
02222 0.0193 0.01737 0.01298 0.00795
Cumulative Proportion  0.362 0.5541 0.6653 0.73599 0.80162 0.85098 0.89337 0.92018 0.
94240 0.9617 0.97907 0.99205 1.00000
```

Hide

```
#exibe a raíz quadrada dos autovalores
wine.pca$sdev
```

```
 [1] 2.1692972 1.5801816 1.2025273 0.9586313 0.9237035 0.8010350 0.7423128 0.5903367
0.5374755 0.5009017 0.4751722 0.4108165 0.3215244
```

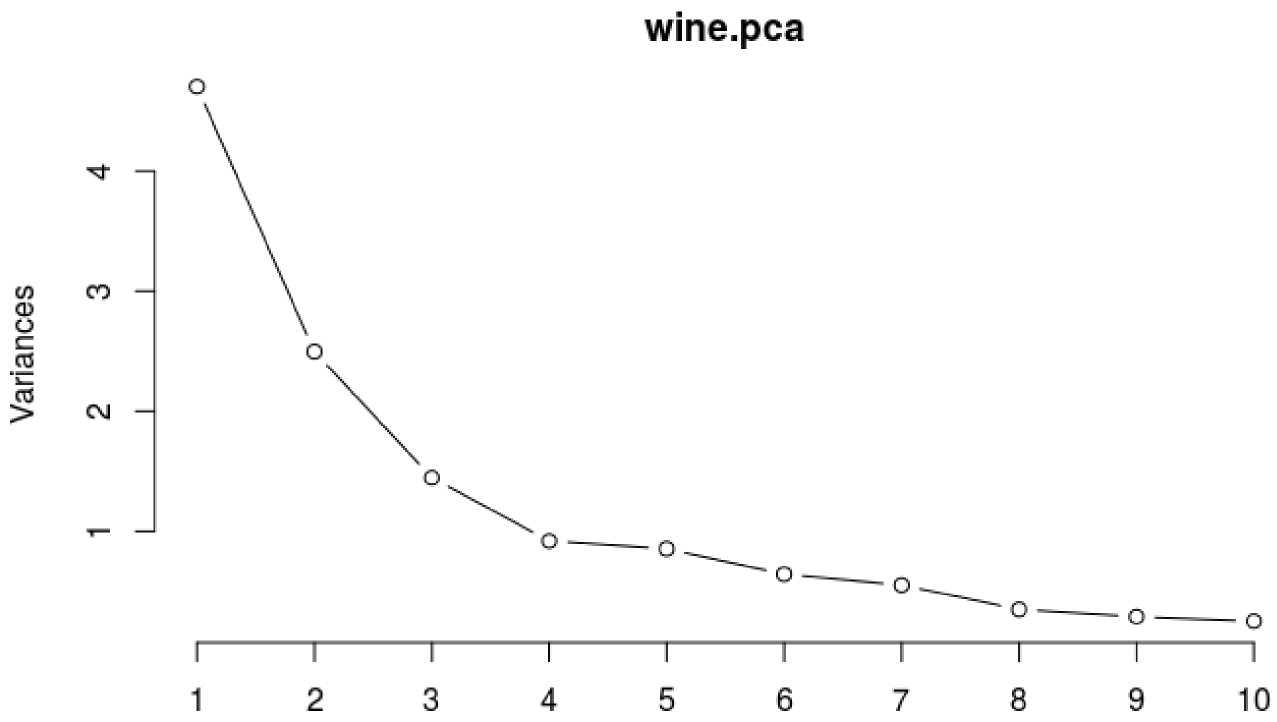
Hide

```
#soma os autovalores
sum((wine.pca$sdev)^2)
```

```
[1] 13
```

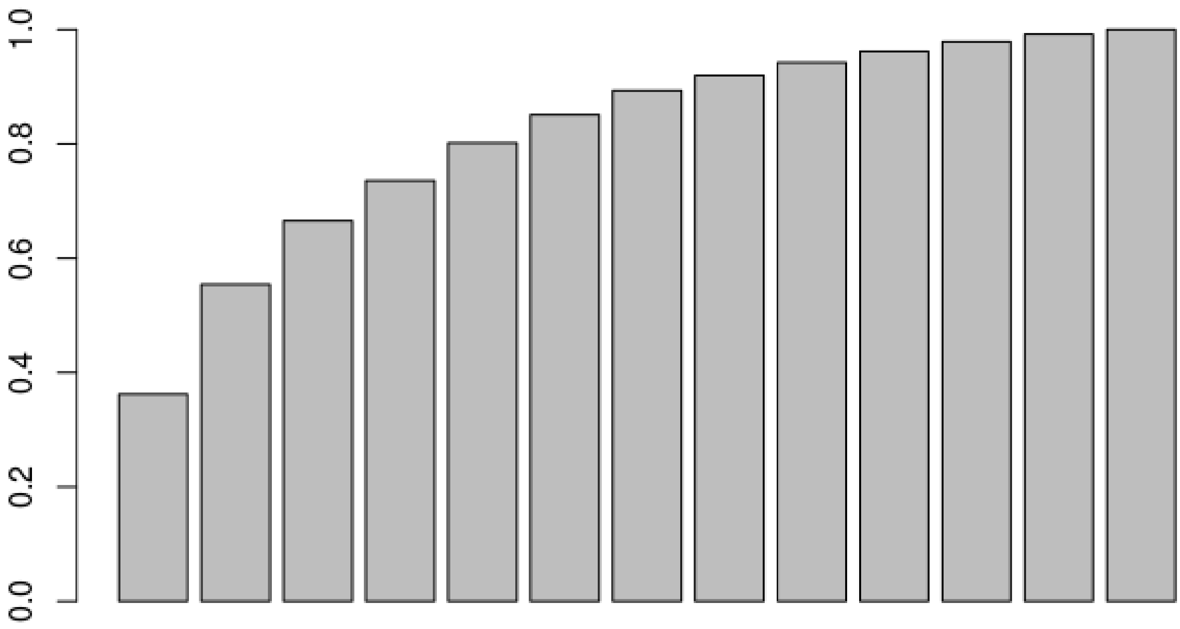
Hide

```
#???
screeplot(wine.pca, type="lines")
```



Hide

```
# Barplot das variancias acumuladas
barplot(cumsum(wine.pca$sdev^2)/sum(wine.pca$sdev^2))
```



Hide

```
# os dois primeiros PCA's explicam aprox 60% da variancia total
# os 5 primeiros explicam aprox 80%
# Os autovetores
dim(wine.pca$rot)
```

```
[1] 13 13
```

Hide

```
# A matriz de autovetores é 13x13
# O 1o autovetor
wine.pca$rot[,1]
```

```
          V2           V3           V4           V5           V6           V7 
   V8           V9          V10          V11          V12          V13 
-0.144329395  0.245187580  0.002051061  0.239320405 -0.141992042 -0.394660845 -0.4229
34297  0.298533103 -0.313429488  0.088616705 -0.296714564 -0.376167411 
         V14 
-0.286752227 
```

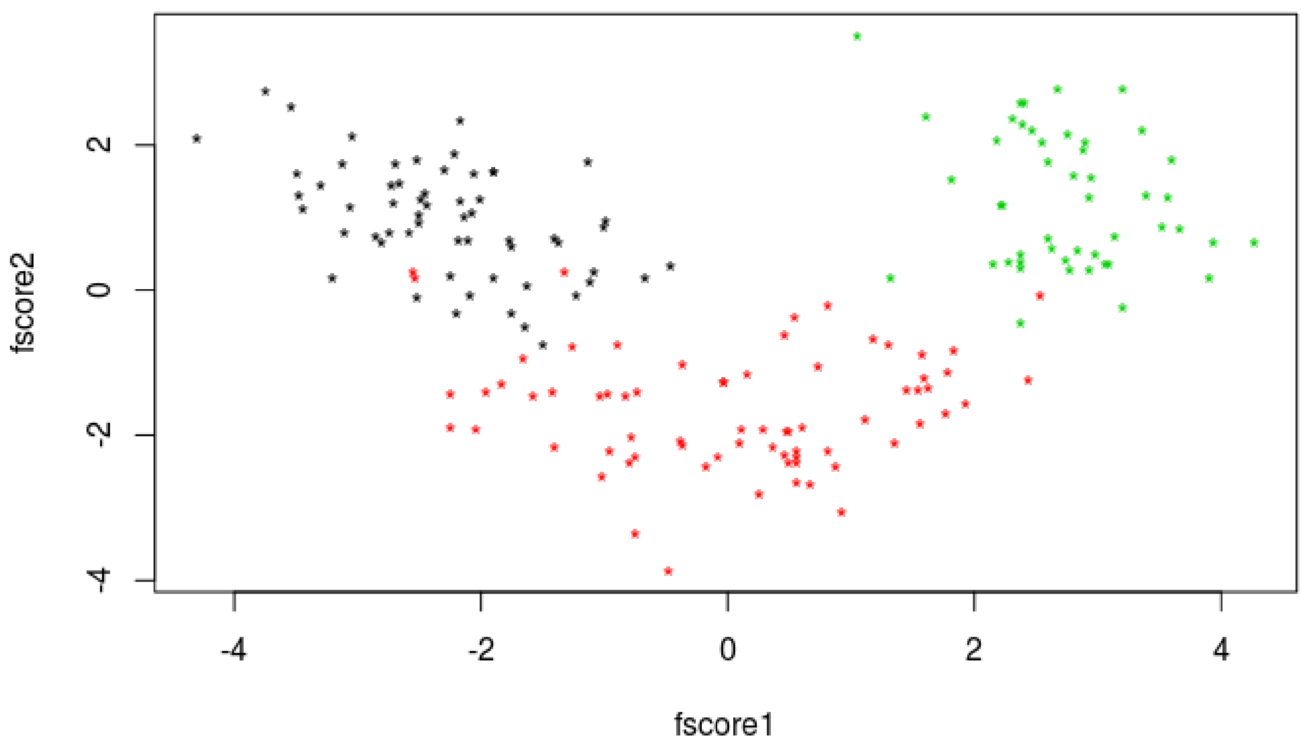
Hide

```
# O 2o autovetor
wine.pca$rot[,2]
```

```
         V2          V3          V4          V5          V6          V7
   V8          V9         V10         V11         V12         V13
 0.483651548  0.224930935  0.316068814 -0.010590502  0.299634003  0.065039512 -0.0033
59812  0.028779488  0.039301722  0.529995672 -0.279235148 -0.164496193
        V14
 0.364902832
```

Hide

```
# Coordenadas dos pontos ao longo do primeiro componente
fscore1 = wine.pca$x[,1]
# Coordenadas dos pontos ao longo do segundo componente
fscore2 = wine.pca$x[,2]
# plot dos pontos projetados
plot(fscore1, fscore2, pch="*", col=wine[,1]+8)
```



Hide

```
#matriz de dados padronizada:
z = scale(wine[2:14])
zMeans <- round(apply(z, 2, mean), 5) # media das colunas da matriz
zSDs <- round(apply(z, 2, sd), 5) # sd das colunas da matriz
```

a)

Hide

```
#os valores das (??) são simplesmente as componentes dos autovetores:
sprintf("Os coeficientes que multiplicam Z para formar Yi1 são: ")
```

```
[1] "Os coeficientes que multiplicam Z para formar Yi1 são: "
```

Hide

```
wine.pca$rot[,1]
```

```
         V2           V3           V4           V5           V6           V7 
   V8           V9          V10          V11          V12          V13 
-0.144329395  0.245187580  0.002051061  0.239320405 -0.141992042 -0.394660845 -0.4229
34297  0.298533103 -0.313429488  0.088616705 -0.296714564 -0.376167411 
        V14 
-0.286752227 
```

Hide

```
sprintf("Os coeficientes que multiplicam Z para formar Yi2 são: ")
```

```
[1] "Os coeficientes que multiplicam Z para formar Yi2 são: "
```

Hide

```
wine.pca$rot[,2]
```

```
         V2           V3           V4           V5           V6           V7 
   V8           V9          V10          V11          V12          V13 
 0.483651548  0.224930935  0.316068814 -0.010590502  0.299634003  0.065039512 -0.0033
59812  0.028779488  0.039301722  0.529995672 -0.279235148 -0.164496193 
        V14 
 0.364902832 
```

b)

Podemos utilizar as regiões superior esquerda, inferior central e superior direita para classificarmos as amostras de vinhos.

c)

Hide

```
x = c(13.95, 3.65, 2.25, 18.4, 90.18, 1.55, 0.48, 0.5, 1.34, 10.2, 0.71, 1.48, 587.14
 )
wineMeans <- round(apply(wine[, 2:14], 2, mean), 5)
wineSDs <- round(apply(wine[, 2:14], 2, sd), 5)
zx <- (x - wineMeans) / (wineSDs)
Yx <- c(wine.pca$rot[,1] %*% zx, wine.pca$rot[,2] %*% zx )
sprintf("No espaço formado pelas componentes principais do conjunto de dados, as cord
enadas do vinho x são: ")
```

```
[1] "No espaço formado pelas componentes principais do conjunto de dados, as cordenad
as do vinho x são: "
```

Hide

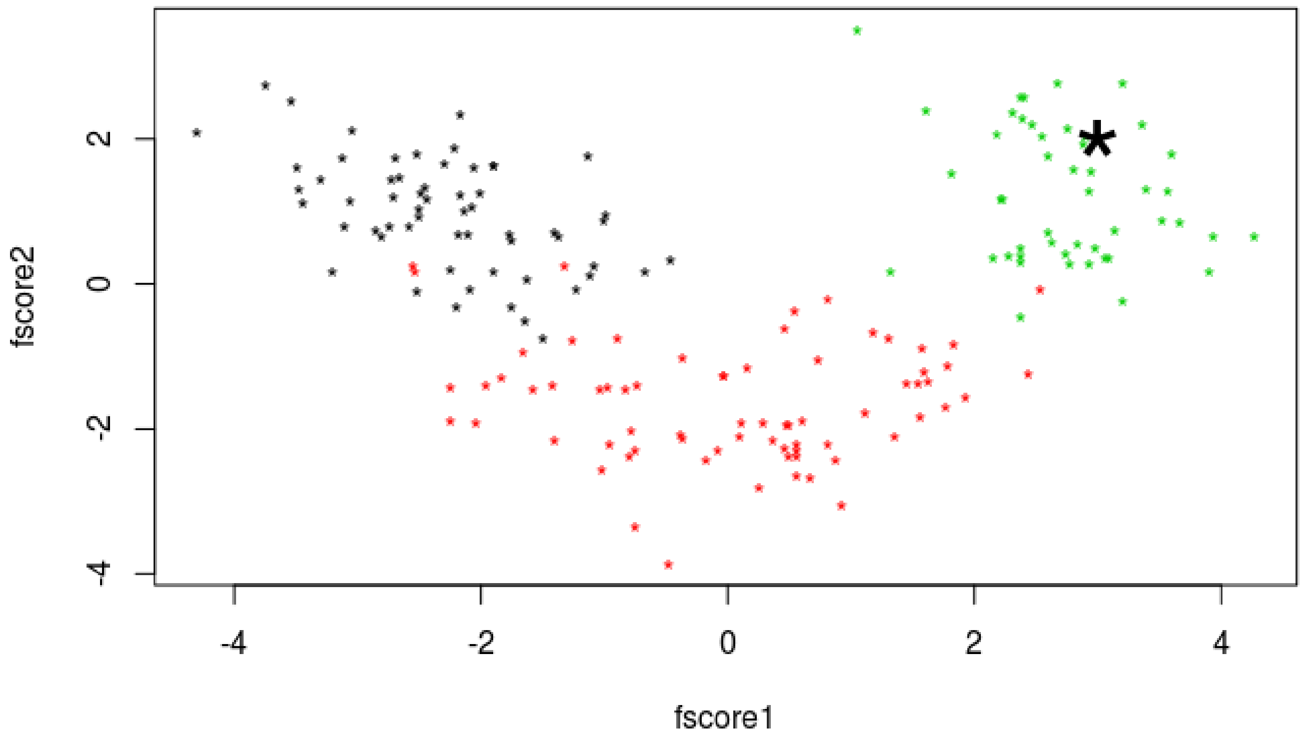
```
Yx
```

```
[1] 2.992734 1.996126
```

Plotando este ponto no gráfico, temos:

Hide

```
plot(fscore1, fscore2, pch="*", col=wine[,1]+8)
points(Yx[1], Yx[2], pch="*", cex=4)
```



Percebemos portanto que o vinho tem grandes chances de pertencer ao cultivar 3

Questão 5

Hide

```
beer <- read.table("Beer.txt", header=TRUE)
head(beer)
```

	COST <int>	SIZE <int>	ALCOHOL <int>	REPUTAT <int>	COLOR <int>	AROMA <int>	TASTE <int>	SES <int>	GROUP <int>
1	90	80	70	20	50	70	60	2	1
2	75	95	100	50	55	40	65	1	1
3	10	15	20	85	40	30	50	4	2
4	100	70	50	30	75	60	80	3	2
5	20	10	25	35	30	35	45	4	1
6	50	100	100	30	90	75	100	3	1

6 rows

Hide

```
summary(beer)
```

```
      COST            SIZE           ALCOHOL          REPUTAT           COLOR     
    AROMA           TASTE             SES            GROUP      
 Min.   :  0.00   Min.   :  0.00   Min.   : 10.00   Min.   :  0.00   Min.   : 0.00  
Min.   : 0.00   Min.   : 25.00   Min.   :0.000   Min.   :1.000  
 1st Qu.: 15.00   1st Qu.: 15.00   1st Qu.: 20.00   1st Qu.: 30.00   1st Qu.:30.00  
1st Qu.:27.50   1st Qu.: 50.00   1st Qu.:2.000   1st Qu.:1.000  
 Median : 50.00   Median : 35.00   Median : 35.00   Median : 40.00   Median :50.00  
Median :45.00   Median : 65.00   Median :3.000   Median :1.000  
 Mean   : 48.81   Mean   : 45.71   Mean   : 49.05   Mean   : 48.33   Mean   :49.52  
Mean   :44.75   Mean   : 65.95   Mean   :3.333   Mean   :1.429  
 3rd Qu.: 80.00   3rd Qu.: 80.00   3rd Qu.: 70.00   3rd Qu.: 65.00   3rd Qu.:75.00  
3rd Qu.:66.25   3rd Qu.: 90.00   3rd Qu.:5.000   3rd Qu.:2.000  
 Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :95.00  
Max.   :90.00   Max.   :100.00   Max.   :8.000   Max.   :2.000  

NA's   :11
```

Hide

```
S <- var(beer[,1:7], na.rm = T)
```

```
the condition has length > 1 and only the first element will be used
```

Hide

```
S
```

```
              COST        SIZE     ALCOHOL   REPUTAT       COLOR       AROMA       TASTE
COST    1174.02397  961.49543  847.979452 -336.3413   16.57534   -40.87329  -52.802511
SIZE     961.49543 1137.92237  982.968037 -320.3311  162.23744    85.01142   20.970320
ALCOHOL  847.97945  982.96804 1039.977169 -361.5183   62.53425    36.79224    9.166667
REPUTAT -336.34132 -320.33105 -361.518265  585.8505 -241.84932 -276.69521 -258.487443
COLOR     16.57534  162.23744   62.534247 -241.8493  722.28311  630.61644  585.410959
AROMA    -40.87329   85.01142   36.792237 -276.6952  630.61644  666.71804  541.775114
TASTE    -52.80251   20.97032    9.166667 -258.4874  585.41096  541.77511  581.329909
```

Hide

```
sqrt(diag(S)) # sd's not very different
```

```
    COST     SIZE  ALCOHOL  REPUTAT    COLOR    AROMA    TASTE 
34.26403 33.73311 32.24868 24.20435 26.87533 25.82088 24.11078
```

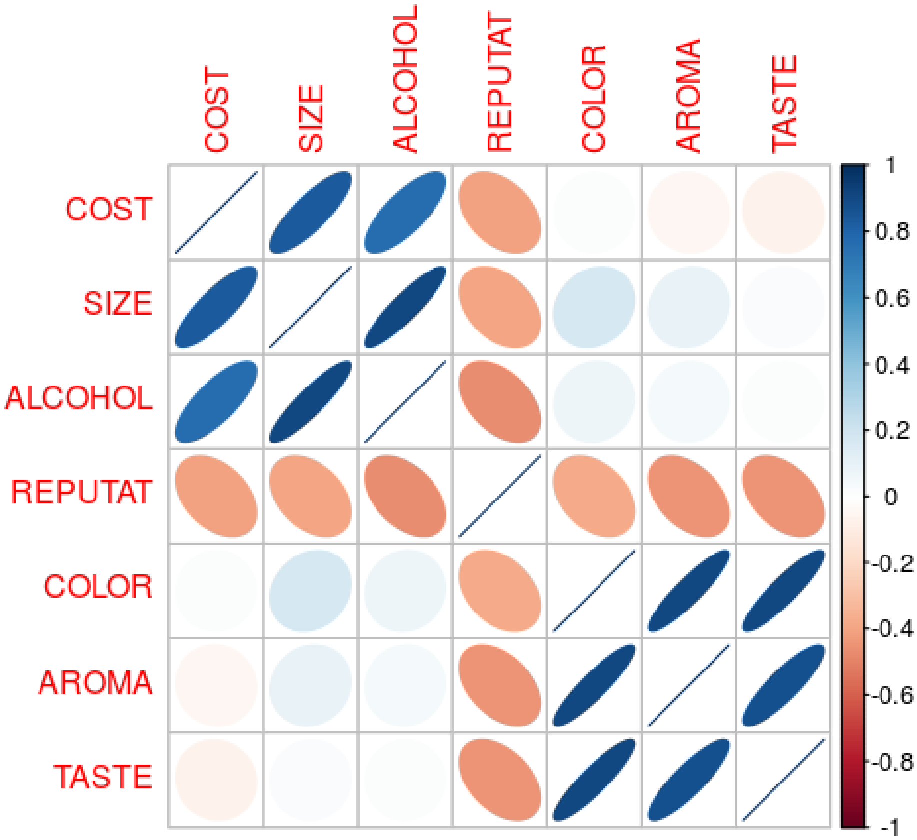
Hide

```
R = cor(beer[,1:7], use ="complete.obs")
round(100*R)
```

```
        COST SIZE ALCOHOL REPUTAT COLOR AROMA TASTE
COST     100   83      77     -41     2    -5    -6
SIZE      83  100      90     -39    18    10     3
ALCOHOL   77   90     100     -46     7     4     1
REPUTAT  -41  -39     -46     100   -37   -44   -44
COLOR      2   18       7     -37   100    91    90
AROMA     -5   10       4     -44    91   100    87
TASTE     -6    3       1     -44    90    87   100
```

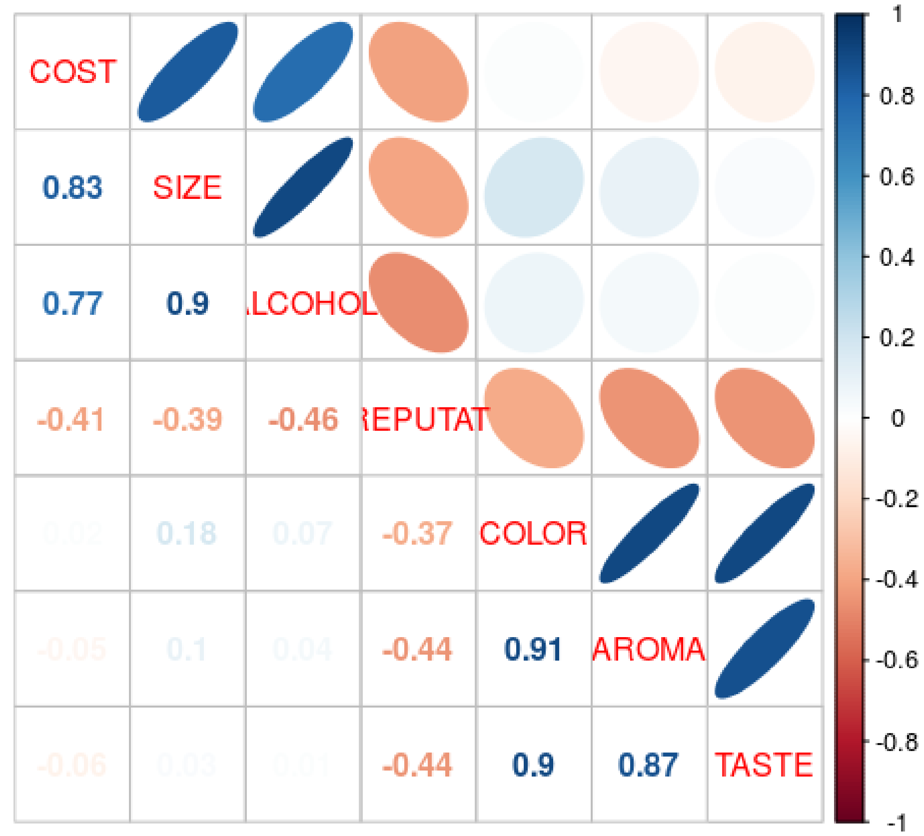
Hide

```
library(corrplot)
corrplot(R, method = "ellipse")
```



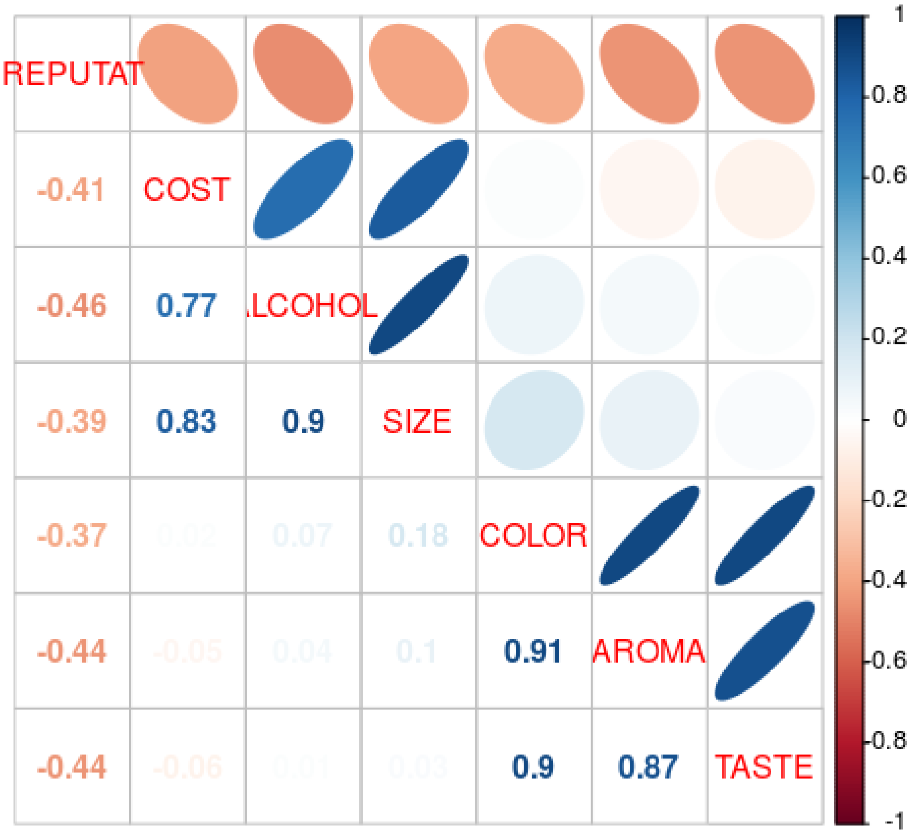
Hide

```
# plotando as elipses e os valores das correlacoes
corrplot.mixed(R, upper = "ellipse")
```



Hide

```
# rearranjando as linhas e colunas para agrupar variaveis com correlacoes parecidas
corrplot.mixed(R, order = "AOE", upper = "ellipse", cl.align = "r")
```



Hide

```
newbeer = na.omit(beer)
S = cov(newbeer[,1:7])
fit = eigen(S) # usa o algoritmo QR em cima da matriz S
# autovalores
fit$values
```

```
[1] 3153.94570 1924.88342  367.94620  267.45829   95.05990   69.65392   29.15759
```

Hide

```
# autovetores
fit$vectors
```

```
            [,1]        [,2]         [,3]        [,4]        [,5]       [,6]        
[,7]
[1,] -0.54635061  0.18490207  0.046480185  0.77610827 -0.16118800 -0.1808547 -0.06416
993
[2,] -0.57392846  0.08254619 -0.332578033 -0.18476541  0.48119230  0.3749770  0.38326
503
[3,] -0.53350251  0.10998646 -0.018418788 -0.58416120 -0.45702058 -0.2852808 -0.26728
556
[4,]  0.24598171  0.22119981 -0.889570411  0.08795166 -0.23288917 -0.1826818  0.06231
067
[5,] -0.11971947 -0.56786380 -0.297723390  0.10330327  0.08273793  0.3143310 -0.67693
250
[6,] -0.09071797 -0.55375799 -0.083017712 -0.01229602  0.31927235 -0.7275381  0.21640
865
[7,] -0.06642820 -0.51851111 -0.005051642  0.06095191 -0.60875842  0.2895003  0.51826
213
```

Hide

```
pca.beer = prcomp(newbeer[,1:7])
# Se quiser obter PCA da matriz de correla\c{c}\~{a}o, use
# pca.beer = prcomp(newbeer[,1:7], scale. = TRUE)
# Os 7 autovetores
pca.beer$rot
```

```
                PC1         PC2          PC3         PC4         PC5        PC6         
   PC7
COST    -0.54635061 -0.18490207  0.046480185 -0.77610827 -0.16118800  0.1808547  0.06
416993
SIZE    -0.57392846 -0.08254619 -0.332578033  0.18476541  0.48119230 -0.3749770 -0.38
326503
ALCOHOL -0.53350251 -0.10998646 -0.018418788  0.58416120 -0.45702058  0.2852808  0.26
728556
REPUTAT  0.24598171 -0.22119981 -0.889570411 -0.08795166 -0.23288917  0.1826818 -0.06
231067
COLOR   -0.11971947  0.56786380 -0.297723390 -0.10330327  0.08273793 -0.3143310  0.67
693250
AROMA   -0.09071797  0.55375799 -0.083017712  0.01229602  0.31927235  0.7275381 -0.21
640865
TASTE   -0.06642820  0.51851111 -0.005051642 -0.06095191 -0.60875842 -0.2895003 -0.51
826213
```

Hide

```
# Os 7 autovalores
(pca.beer$sdev)^2
```

```
[1] 3153.94570 1924.88342  367.94620  267.45829   95.05990   69.65392   29.15759
```

Hide

```
#verificando que os autovetores gerados por prcomp são os mesmos gerados por eigen:
round(abs(pca.beer$rot) - abs(fit$vectors), 2)
```

```
        PC1 PC2 PC3 PC4 PC5 PC6 PC7
COST      0   0   0   0   0   0   0
SIZE      0   0   0   0   0   0   0
ALCOHOL   0   0   0   0   0   0   0
REPUTAT   0   0   0   0   0   0   0
COLOR     0   0   0   0   0   0   0
AROMA     0   0   0   0   0   0   0
TASTE     0   0   0   0   0   0   0
```

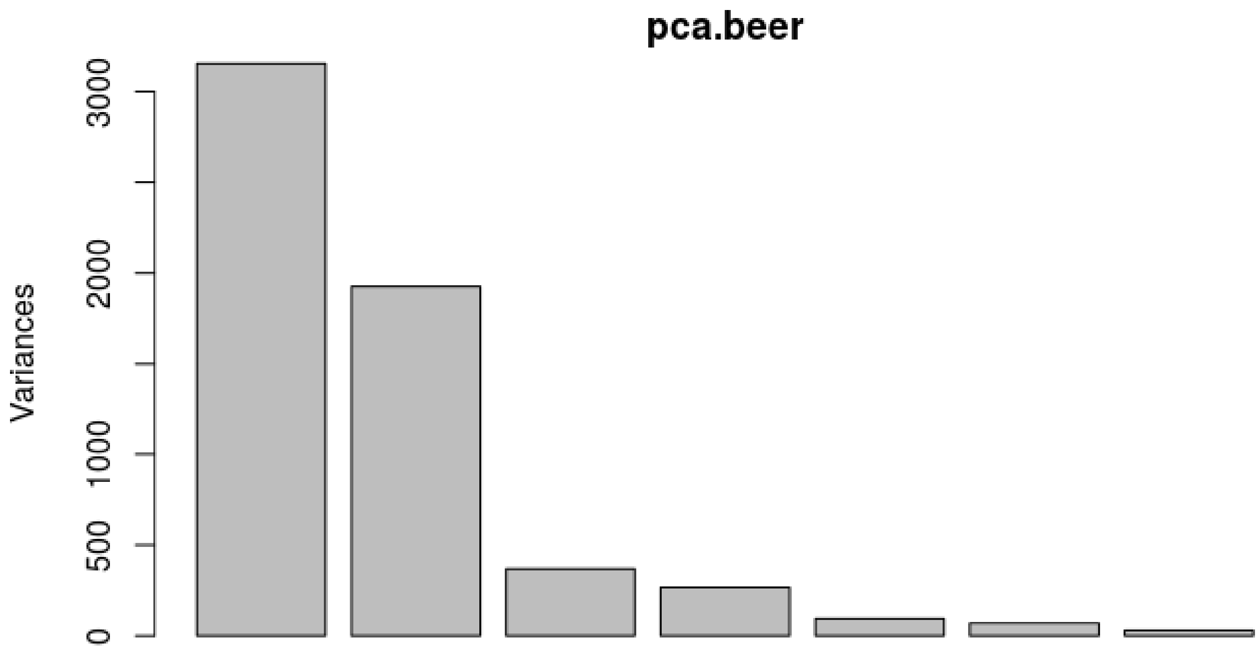
Hide

```
# verifique que os autovetores tem norma euclidiana = 1.
# Por exemplo, o 1o PCA:
sum(pca.beer$rot[,1]^2)
```

```
[1] 1
```

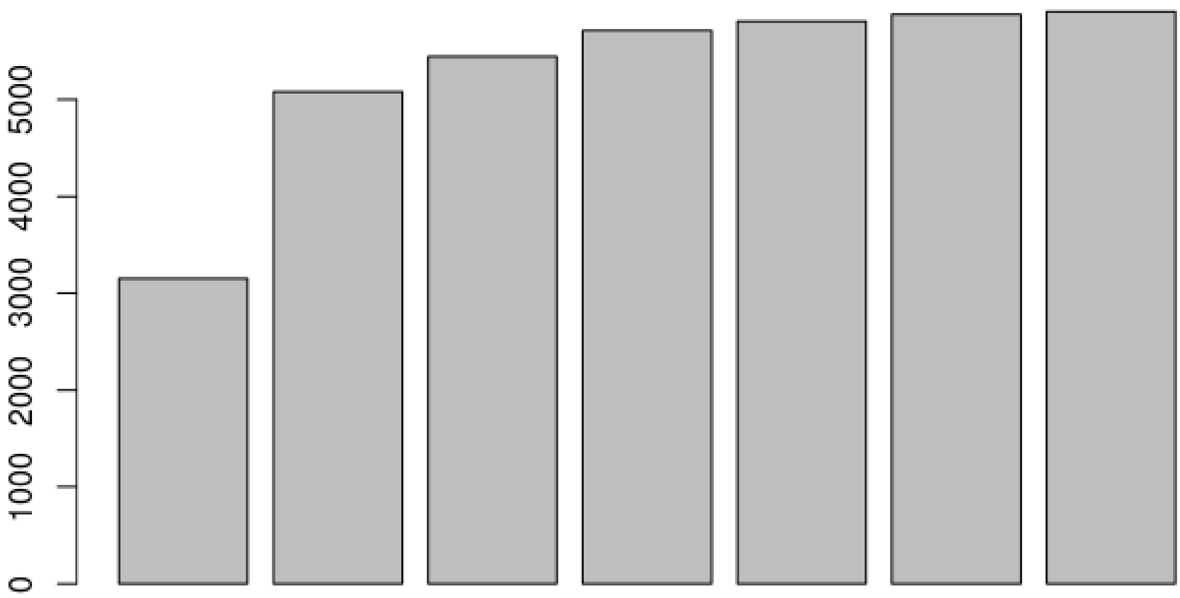
Hide

```
# Grafico scree com os 7 autovalores (ou variancias de cada PCA)
plot(pca.beer)
```



Hide

```
# Barplot das variancias acumuladas indicando a escolha de 2 PCAs
barplot(cumsum(pca.beer$sdev^2))
```



Hide

```
# Resumo
summary(pca.beer)
```

```
Importance of components:
                           PC1     PC2      PC3      PC4     PC5     PC6     PC7
Standard deviation     56.1600 43.8735 19.18192 16.35415 9.74987 8.34589 5.39978
Proportion of Variance  0.5338  0.3258  0.06228  0.04527 0.01609 0.01179 0.00494
Cumulative Proportion   0.5338  0.8596  0.92192  0.96719 0.98328 0.99506 1.00000
```

Hide

```
# Note que o quadrado da linha Standard deviation acima eh igual aos autovalores
# obtidos com fit$values
round(sum(fit$values) - sum(pca.beer$sdev^2),2)
```

```
[1] 0
```

Hide

```
# Vamos usar apenas os dois 1os PCs para representar R com dois fatores
# Carga do Fator = sqrt(LAMBDA) * EIGENVECTOR
cargafat1 = pca.beer$sdev[1] * pca.beer$rot[,1]
cargafat2 = pca.beer$sdev[2] * pca.beer$rot[,2]
# matriz de cargas
L = cbind(cargafat1, cargafat2)
rownames(L) = rownames(R)[1:7]
round(L, 2)
```

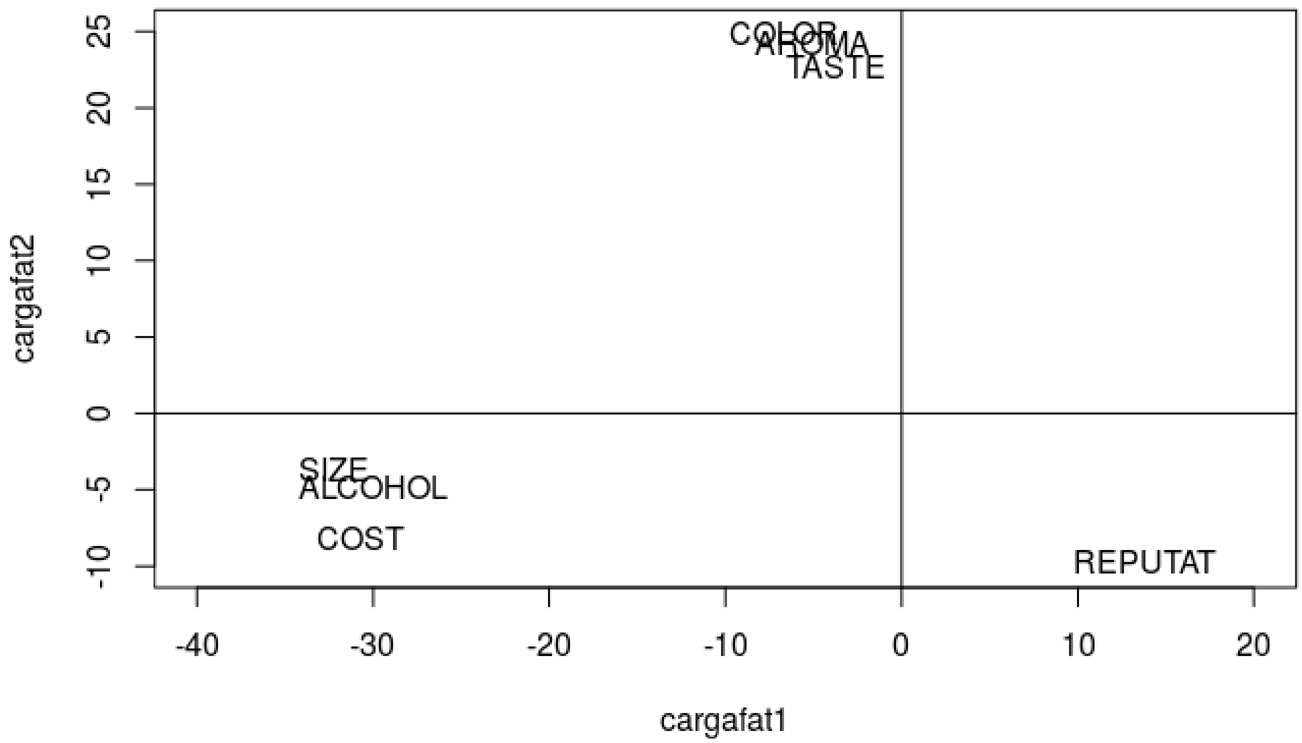
```
        cargafat1 cargafat2
COST       -30.68     -8.11
SIZE       -32.23     -3.62
ALCOHOL    -29.96     -4.83
REPUTAT     13.81     -9.70
COLOR       -6.72     24.91
AROMA       -5.09     24.30
TASTE       -3.73     22.75
```

Hide

```
plot(L, type="n",xlim=c(-40, 20), ylim=c(-10, 25))
text(L, rownames(L))
```

Hide

```
abline(h=0)
abline(v=0)
```

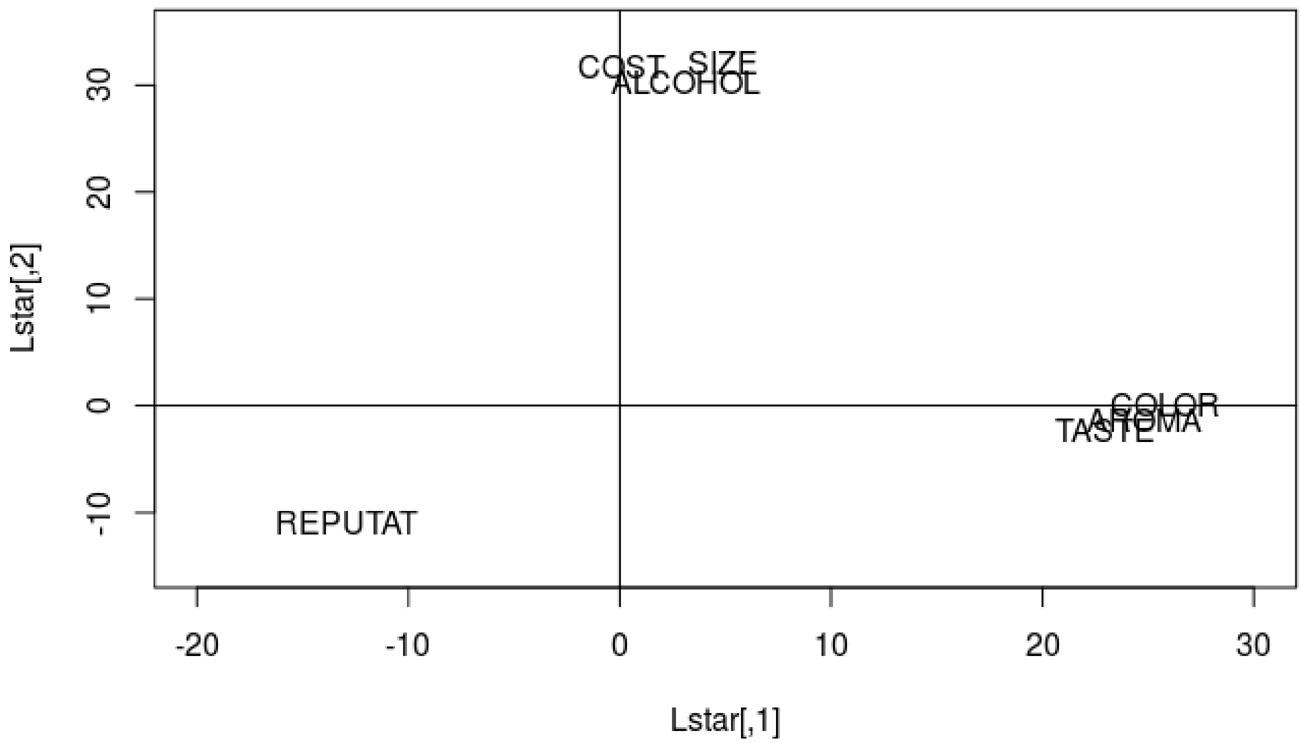


Hide

```
# Fazendo manualmente uma rotacao horaria de pi/2+15*pi/180
phi = pi/2 + 15*(pi/180)
T = matrix(c(cos(phi), -sin(phi), sin(phi), cos(phi)), ncol=2, byrow=T)
Lstar = L %*% T # usando a multiplicacao por linha da matriz L
plot(Lstar, type="n", xlim=c(-20, 30), ylim=c(-15, 35))
text(Lstar, rownames(L))
```

Hide

```
abline(h=0); abline(v=0)
```



Hide

```
round(Lstar,2)
```

```
           [,1]   [,2]
COST       0.11  31.74
SIZE       4.84  32.07
ALCOHOL    3.09  30.19
REPUTAT -12.95 -10.83
COLOR     25.81   0.05
AROMA     24.79  -1.37
TASTE     22.94  -2.28
```

Hide

```
matpsi = diag(diag(S - Lstar %*% t(Lstar)))
round(matpsi, 2)
```

```
       [,1]  [,2] [,3]   [,4]  [,5] [,6] [,7]
[1,] 166.76  0.00    0   0.00  0.00  0.0  0.0
[2,]   0.00 85.92    0   0.00  0.00  0.0  0.0
[3,]   0.00  0.00  119   0.00  0.00  0.0  0.0
[4,]   0.00  0.00    0 300.83  0.00  0.0  0.0
[5,]   0.00  0.00    0   0.00 56.36  0.0  0.0
[6,]   0.00  0.00    0   0.00  0.00 50.5  0.0
[7,]   0.00  0.00    0   0.00  0.00  0.0 49.9
```

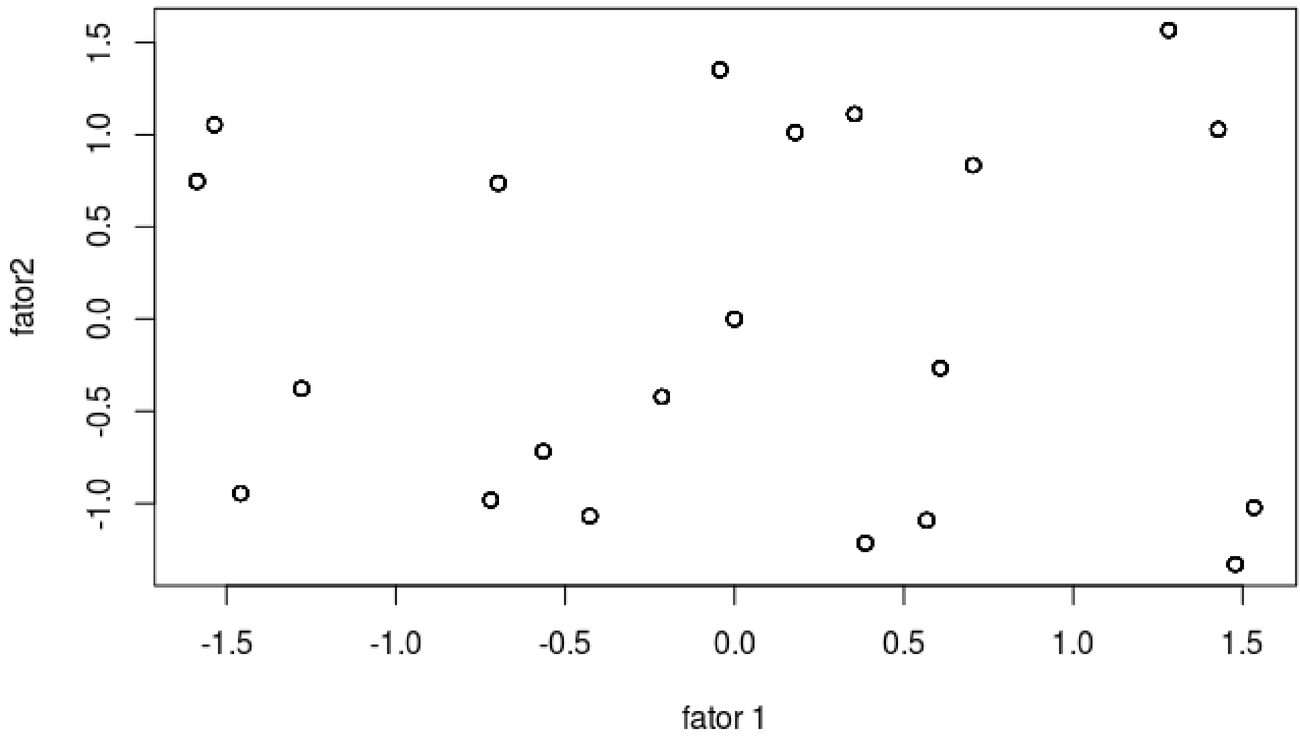
Hide

```
sum( (S - Lstar %*% t(Lstar) - matpsi)^2 )/sum(S^2)
```

```
[1] 0.005303857
```

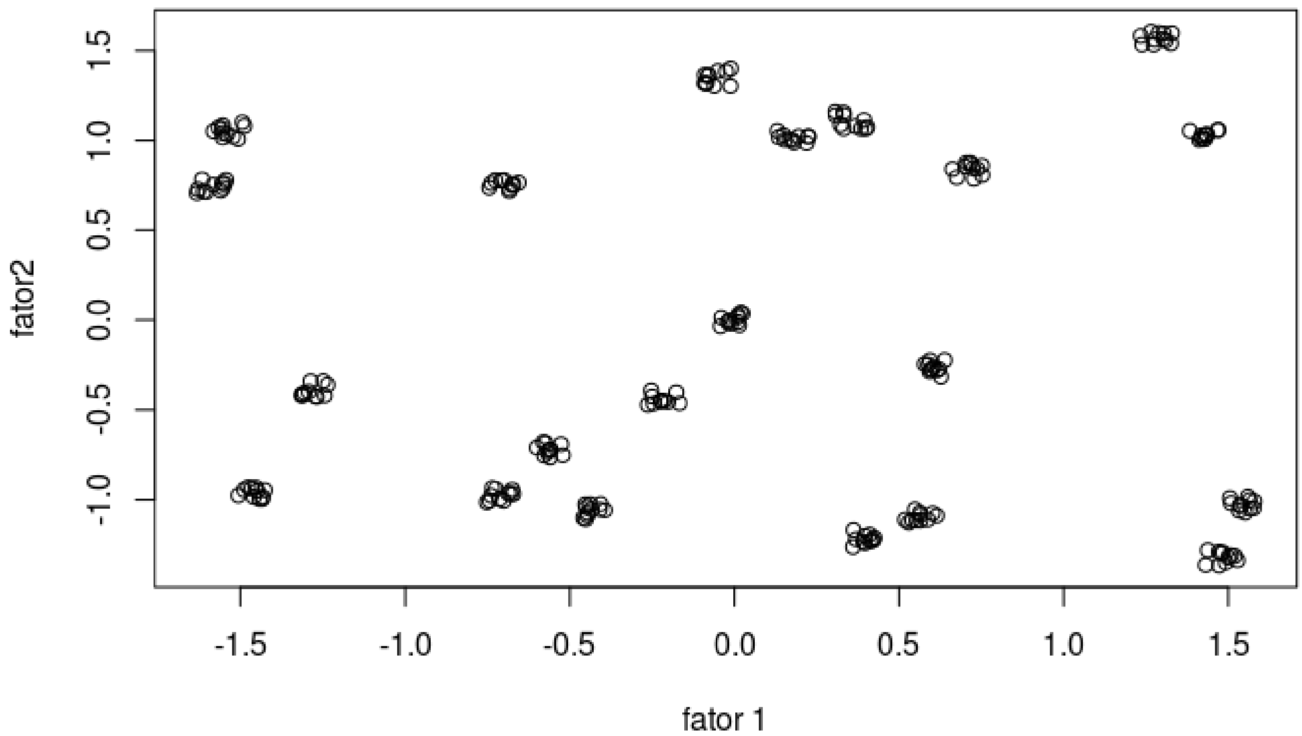
Hide

```
## Factor scores dos n=220 individuos
factors <- matrix(0, nrow=nrow(beer), ncol=2)
mu <- apply(newbeer[,1:7], 2, mean)
for(i in 1:nrow(newbeer)){
  y <- as.numeric(newbeer[i, 1:7] - mu)
  factors[i,] <- lm(y~0 + Lstar)$coef
}
plot(factors, xlab="fator 1", ylab="fator2")
```



Hide

```
# mas... onde estao os 220 individuos?
# Varios individuos poduziram o MESMO vator x --> estimamos com os mesmos fatores
plot(jitter(factors, amount=0.05), xlab="fator 1", ylab="fator2")
```



fator2
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
fator 1