

Tarefa 2 de Aprendizado não Supervisionado (MO 433)

17/11/2021

Autores

- Fábio Kenji Jojima (RA 232024)
- Lucas Alves Racoci (RA 156331)

Objetivo

Encontrar um valor adequado de números de componentes (clusters) para dividir os [dados](#) [1] fornecidos pelo professor e comparar diferentes medidas de qualidade para essa divisão.

Metodologia

Para realizar esse objetivo, usamos a linguagem R, seguindo os seguintes passos:

- Leitura dos dados
- Clusterização por K Médias (K Means)
 - Execução do método k-means para $k = 2$ e mensuração por silhueta
 - Generalização da execução para k de 2 a 15
- Escolha de um valor de k
- Clusterização por Mistura de Modelos Gaussianos (GMM)
 - Determinação das restrições do modelo
 - Mensuração por BIC
 - Mensuração por ICL
- Comparação dos diferentes modelos de GMM

Estes passos serão detalhados a seguir. Para mais detalhes sobre o script completo executado, veja o apêndice [3]. Os resultados mostrados neste documento podem diferir dos resultados executando novamente o script pois várias das funções utilizam valores pseudo aleatórios para serem executadas. Para se obter uma maior consistência nos resultados é possível utilizar a função `set.seed` nativa do R.

Leitura dos Dados

Para ler os dados foi utilizado a função `read` nativa do R:

```
url <- "https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/ex2-data.csv"
df <- read.csv(url, header = FALSE, sep = "")
```

Clusterização por K Médias (*K Means*)

Testes Iniciais (K = 2)

Para encontrar os K-Means de 2 clusters foi usada a função `kmeans` nativa do R:

```
> result <- kmeans(df, 2)
```

Para encontrar os valores de distância e então calcular a silhueta dos clusters encontrados, foi utilizada a função `silhouette` da biblioteca "`cluster`". Para instalá-la, usamos o seguinte comando:

```
install.packages("cluster")  
library("cluster")
```

Assim, para obter a silhueta do resultado da clusterização:

```
> sil <- silhouette(result$cluster, dist(df))
```

Através do comando `colnames`, podemos encontrar a coluna contendo os valores desejados:

```
> colnames(sil)
```

```
[1] "cluster"    "neighbor"    "sil_width"
```

Assim, fazemos a média dos valores da terceira coluna para encontrar a silhueta:

```
> mean(sil[, 3])
```

```
[1] 0.1360951
```

Com o valor da silhueta para $k = 2$, podemos expandir para valores de 2 a 15

Execução para K de 2 a 15

Utilizando os testes feitos para $k = 2$ como base, foram feitas iterações para k variando de 2 a 15 e armazenando os resultados das silhuetas encontradas em `ss`, um vetor coluna inicializado:

```
ss <- c() # create empty vector  
for (k in 2:15) {  
  result <- kmeans(df, k)  
  sil <- silhouette(result$cluster, dist(df))  
  s_mean <- mean(sil[, 3])  
  ss <- append(ss, s_mean)  
}
```

Com os valores das silhuetas, é então gerado o gráfico com a função `plot`. Os valores usados para parametrizar a grade foram escolhidos ad-hoc apenas para tornar a visualização mais clara.

```
plot(
  2:15,
  ss,
  type = "b",
  col="blue",
  xlab = "Numbers of clusters K",
  ylab = "Silhouette"
)
axis(side=1, at=c(2:15))
abline(
  h=seq(0.16, 0.24, len=5),
  v=seq(2, 15, len=14),
  col="grey"
)
```

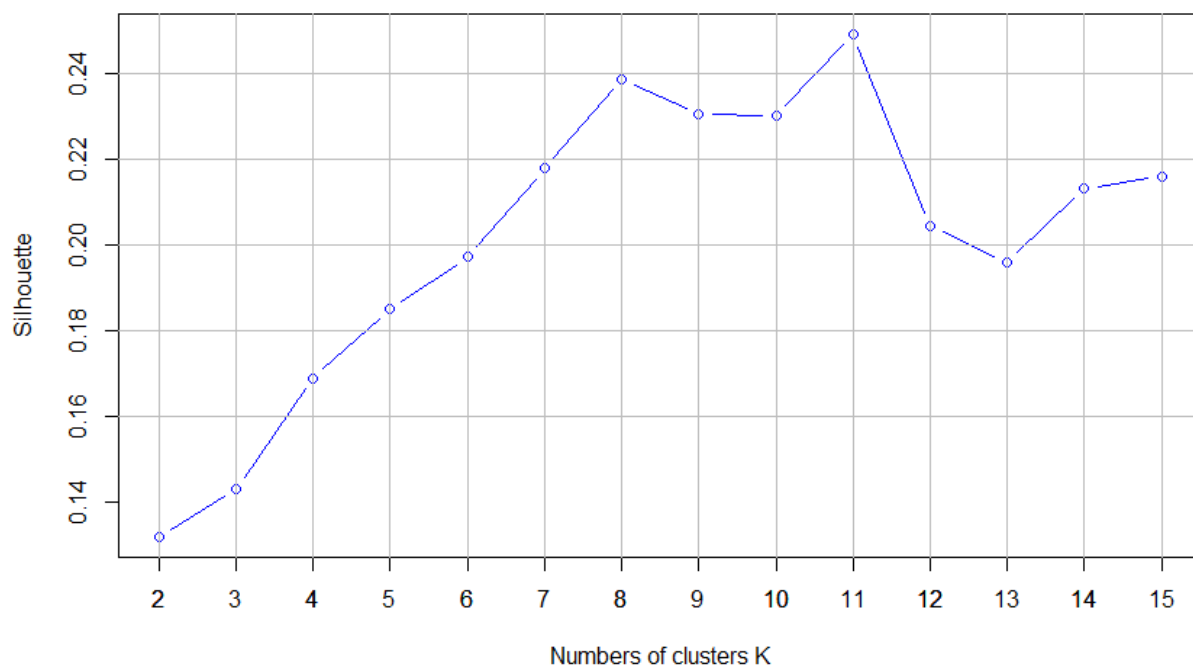


Gráfico 1

Outra forma de medir a qualidade interna da clusterização apresentada em aula é a métrica GAP. Utilizando a função `clusGap` também pertencente a biblioteca "`cluster`", foi possível obter os valores desejados.

```
gap_stat <- clusGap(
  df,
  FUN = kmeans,
  nstart = 1,
```

```
K.max = 15,  
B = 50  
)
```

Então utilizando a função `fviz_gap_stat` da biblioteca **"factoextra"**, encontramos o primeiro K ótimo para o valores encontrados anteriormente:

```
install.packages("factoextra")  
library("factoextra")  
  
fviz_gap_stat(gap_stat)  
abline(  
  h=seq(0.45, 0.70, len=6),  
  v=seq(2, 15, len=14),  
  col="grey"  
)
```

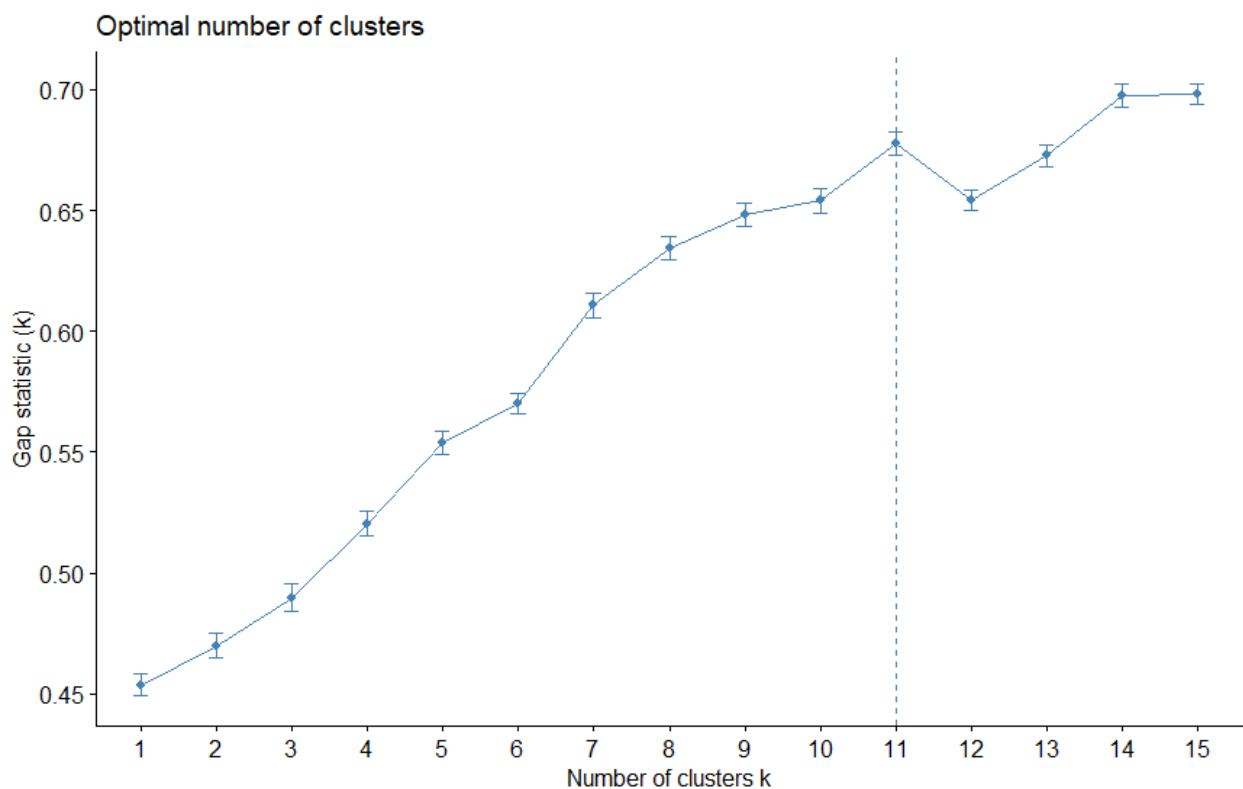


Gráfico 2

Dos resultados encontrados no [gráfico 1](#) gerado com os valores de silhueta, achamos que 11 seria o melhor número de clusters, aplicando GAP como alternativa e demonstrado no [gráfico 2](#) também encontramos que o número ótimo de clusters seria 11.

Clusterização por Mistura de Modelos Gaussianos (*GMM*)

Para executar o algoritmo de Mistura de Modelos Gaussianos (GMM) usamos a biblioteca **"mclust"** conforme sugerido em aula.

```
install.packages("mclust")  
library("mclust")
```











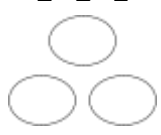



Para usar essa biblioteca foi necessário identificar, na linguagem de modelos do **"mclust"**, os nomes das restrições solicitadas. Na convenção usada pela biblioteca, os modelos são identificados por três letras, onde a primeira indica a restrição de volume, a segunda indica a restrição de formato e a terceira indica restrição de orientação.

As restrições podem também de três tipos "V", "I", "E", com os seguintes significados para uma restrição específica:

- "V" significa que todas as gaussianas podem variar livremente quanto a restrição
- "E" significa que todas as gaussianas devem ser iguais quanto a restrição
- "I" significa que a restrição deve ser coordenada com o eixo.

A restrição "I" não se aplica a volumes, por isso a primeira letra nunca é "V". Quando a restrição de formato é coordenado ao eixo a restrição de orientação também é, por isso não ocorrem os casos VIV, VIE, EIV, EIE.

Uma ilustração das combinações possíveis pode ser vista na figura a seguir

V V V 	V V I 	V V E 	E V V 	E V I 	E V E 
V E V 	V E I 	V E E 	E E V 	E E I 	E E E 
-	V I I 	-	-	E I I 	-

Nosso objetivo foi analisar e comparar três modelos de restrições:

1. VII: Esféricos sem restrição no tamanho.
2. VVI: Com matrizes de variância diagonais, isto é, com a orientação coordenada ao eixo.
3. VVV: Sem restrições, com volume, formato e orientação variando livremente.

Escolhido os parâmetros para as gaussianas procuradas, é então utilizado a função `mclustBIC` para obter os valores da matriz BIC (*“Bayesian information criterion”*) para seleção dos modelos.

```
BIC <- mclustBIC(df, G = 2:15, modelNames = c("VII", "VVI", "VVV"))
plot(BIC)
```

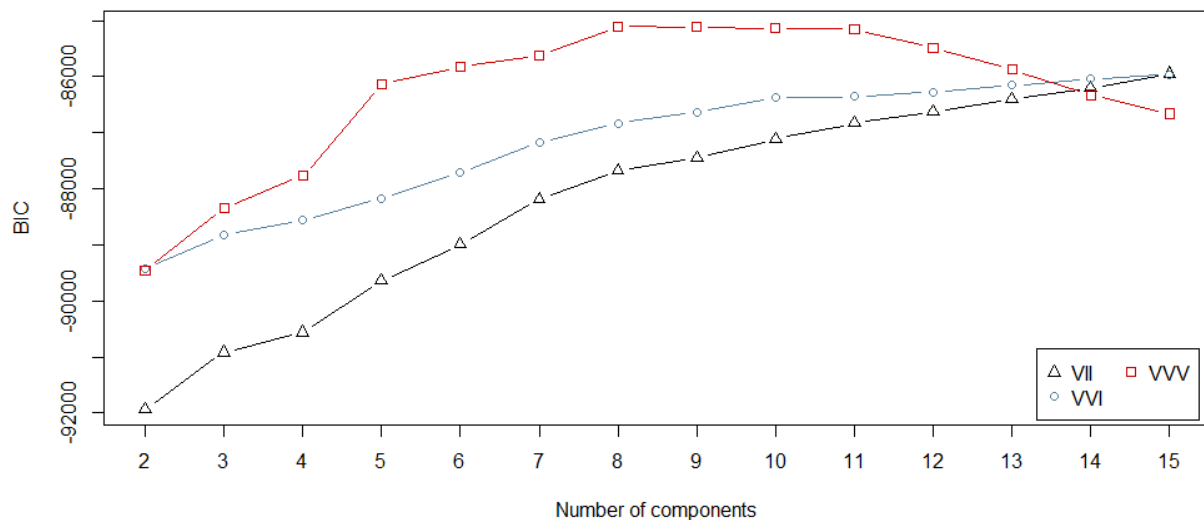


Gráfico 3

Para o k escolhido a partir do k-means, nos passos anteriores (k = 11), são obtidos os valores:

```
> BIC11 <- mclustBIC(df, G = 11:11, modelNames = c("VII", "VVI", "VVV"))
> summary(BIC11)
```

```
Best BIC values:
          VVV,11    VVI,11    VII,11
BIC      -85154.46 -86357.070 -86831.380
BIC diff      0.00  -1202.608  -1676.918
```

Similarmente utilizando a função `mclustICL` foi gerado os valores ICL (*“Integrated Complete-data Likelihood”*) do nosso dataframe para uma melhor comparação na escolha do número de clusters.

```
ICL <- mclustICL(df, G = 2:15, modelNames = c("VII", "VVI", "VVV"))
plot(ICL)
```

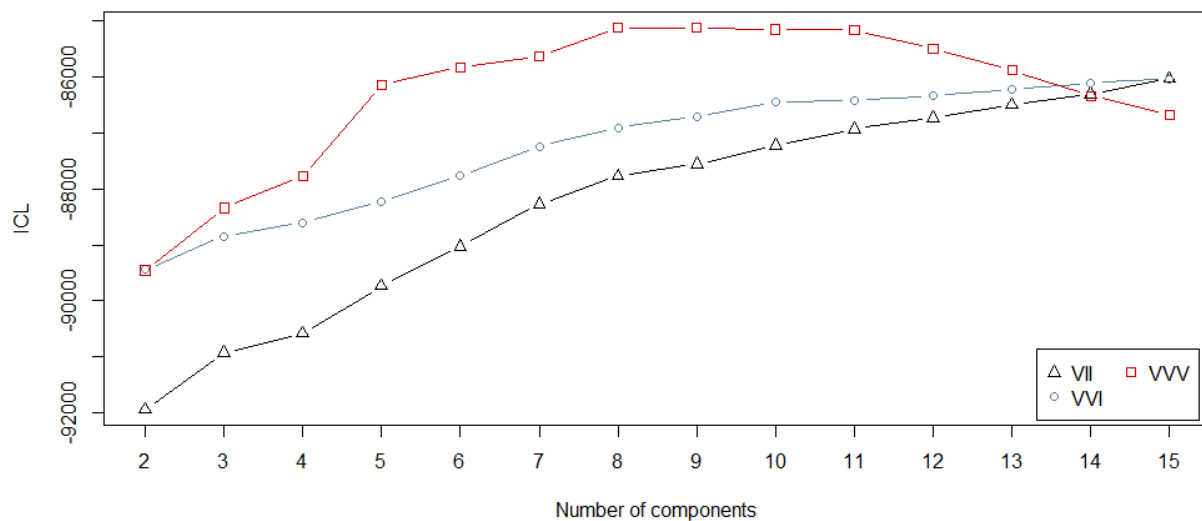


Gráfico 4

Novamente são comparados os valores para $k = 11$:

```
> ICL11 <- mclustICL(df, G = 11:11, modelNames = c("VII", "VVI", "VVV"))
> summary(ICL11)
```

Best ICL values:			
	VVV,11	VVI,11	VII,11
ICL	-85162.8	-86422.423	-86929.702
ICL diff	0.0	-1259.622	-1766.901

Pode-se perceber que os valores obtidos no [gráfico 3](#) e no [gráfico 4](#) são muito parecidos e que o BIC e o ICL selecionam o mesmo modelo, VVV, para os clusters de 8 à 11 contendo os maiores valores.

Para comparar a classificação nos 3 modelos para o k escolhido (11), precisamos obter as classes de cada ponto do dataset, o que pode ser feito pelas linhas:

```
> cVVV <- Mclust(df, G = 11, modelNames = c("VVV"))$classification
> cVVI <- Mclust(df, G = 11, modelNames = c("VVI"))$classification
> cVII <- Mclust(df, G = 11, modelNames = c("VII"))$classification
```

Assim, foi possível comparar o índice de Rand ajustado do modelo de gaussianas esféricas ("VII") com modelo de gaussianas sem restrições ("VVV") através do comando:

```
> adjustedRandIndex(cVII, cVVV)
```

0.6652865

Também comparamos o índice de Rand ajustado do modelo de gaussiana diagonais (“VVI”) com o sem restrições (“VVV”), através do comando:

```
> adjustedRandIndex(cVVI, cVVV)
```

```
0.8285284
```

Como pode ser visto nesses valores, e também é possível ver nos gráficos de BIC e ICL, o modelo de gaussianas diagonais é mais próximo do sem restrições que o de gaussianas esféricas.

Para usar outras métricas, instalamos a biblioteca **"mclustcomp"**. Para mais detalhes sobre essas métricas veja o apêndice [2].

```
install.packages("mclustcomp")  
library("mclustcomp")
```

De forma semelhante, primeiro obtemos as métricas comparando o modelo esférico (“VII”) com o sem restrições (“VVV”) e depois o de gaussianas esféricas com o sem restrições, através dos seguintes comandos. Alinhamos as saídas dos comandos a seguir.

```
> mclustcomp(cVII, cVVV)
```

	types	scores
1	adjrand	6.652865e-01
2	chisq	7.617932e+03
3	f	1.314315e+02
4	fmi	7.022892e-01
5	jaccard	5.397065e-01
6	jent	4.030723e+00
7	mhm	8.480000e-01
8	mi	2.619341e+00
9	mirkin	6.404800e+04
10	mmm	8.480000e-01
11	nmi1	7.878255e-01
12	nmi2	7.877643e-01
13	nvi	2.121132e-01
14	overlap	7.452860e-01
15	pd	4.299270e+05
16	rand	9.358879e-01
17	sdsc	7.010511e-01
18	smc	9.358879e-01
19	tanimoto	5.397065e-01
20	tversky	5.397065e-01
21	vdm	3.040000e+02
22	vi	1.411381e+00
23	wallace1	7.452860e-01
24	wallace2	6.617730e-01

```
> mclustcomp(cVVI, cVVV)
```

	types	scores
1	adjrand	8.285284e-01
2	chisq	8.463362e+03
3	f	1.146956e+02
4	fmi	8.456475e-01
5	jaccard	7.323344e-01
6	jent	3.831666e+00
7	mhm	9.210000e-01
8	mi	2.932308e+00
9	mirkin	3.054600e+04
10	mmm	9.210000e-01
11	nmi1	8.670461e-01
12	nmi2	8.670370e-01
13	nvi	1.329448e-01
14	overlap	8.622098e-01
15	pd	4.424400e+05
16	rand	9.694234e-01
17	sdsc	8.454885e-01
18	smc	9.694234e-01
19	tanimoto	7.323344e-01
20	tversky	7.323344e-01
21	vdm	1.580000e+02
22	vi	8.993585e-01
23	wallace1	8.294034e-01
24	wallace2	8.622098e-01

Novamente, estes valores, em geral, mostram que a clusterização por gaussianas diagonais está mais próxima da sem restrições que a por gaussianas esféricas.

Conclusão

Apesar do GMM não encontrar um valor ótimo, escolhendo o K encontrado utilizando K-means, podemos verificar que o melhor modelo é o “VVV” (gaussianas sem restrições), seguido de “VVI”, e depois “VII”. Assim, ao menos para esses dados, parece que quanto menos restrições, “melhor” a classificação.

Apêndice

[1] Dados disponíveis em :

<https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/ex2-data.csv> (último acesso 16:58 do dia 13/11/2021).

[2] <https://cran.r-project.org/web/packages/mclustcomp/mclustcomp.pdf> (último acesso 23:27 do dia 16/11/2021)

[3] O script completo utilizado:

```
1. install.packages("cluster")
2. install.packages("factoextra")
3. install.packages("mclust")
4. install.packages("mclustcomp")
5.
6. library("cluster")
7. library("factoextra")
8. library("mclust")
9. library("mclustcomp")
10.
11. url <- "https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/ex2-data.csv"
12. df <- read.csv(url, header = FALSE, sep = "")
13.
14. ss <- c()
15. for (k in 2:15) {
16.   result <- kmeans(df, k)
17.   sil <- silhouette(result$cluster, dist(df))
18.   s_mean <- mean(sil[, 3])
19.   ss <- append(ss, s_mean)
20. }
21.
22. plot(
23.   2:15,
24.   ss,
```

```

25. type = "b",
26. col="blue",
27. xlab = "Number of clusters K",
28. ylab = "Silhouette"
29. )
30. axis(side=1, at=c(2:15))
31. abline(
32.   h=seq(0.16, 0.24, len=5),
33.   v=seq(2, 15, len=14),
34.   col="grey"
35. )
36.
37. gap_stat <- clusGap(
38.   df,
39.   FUN = kmeans,
40.   nstart = 1,
41.   K.max = 15,
42.   B = 50
43. )
44.
45. # Marks first max k
46. fviz_gap_stat(gap_stat)
47. abline(
48.   h=seq(0.45, 0.70, len=6),
49.   v=seq(2, 15, len=14),
50.   col="grey"
51. )
52.
53. # GMM analysis
54. BIC <- mclustBIC(df, G = 2:15, modelNames = c("VII", "VI", "V"))
55. ICL <- mclustICL(df, G = 2:15, modelNames = c("VII", "VI", "V"))
56. summary(BIC)
57. summary(ICL)
58.
59. plot(BIC)
60. plot(ICL)
61.
62. cVV <- Mclust(df, G = 11, modelNames = c("V"))$classification
63. cVI <- Mclust(df, G = 11, modelNames = c("VI"))$classification
64. cVII <- Mclust(df, G = 11, modelNames = c("VII"))$classification
65.
66. adjustedRandIndex(cVV, cVI)

```

67. `adjustedRandIndex(cVV, cVII)`

68.

69. `mclustcomp(cVV, cVII)`

70. `mclustcomp(cVV, cVII)`

71.