Agrupamento com K-means

Marcos Vinícius dos Santos Ferreira

2018-04-16

Contents

Introdução

---

k-means é um algoritmo de aprendizado de máquina não supervisionado usado para encontrar grupos de observações (clusters) que compartilham *características semelhantes*. Qual é o significado da aprendizagem não supervisionada? Isso significa que as observações dadas no conjunto de dados não são rotuladas, não há resultado a ser previsto.

Para o experimento, o desafio é usar o conjunto de dados de bebidas para prever onde as pessoas bebem mais cerveja, vinho e bebidas espirituosas?

Carregamento dos dados

```
# lendo o dataset
data.drinks = read.csv('../../data/alcohol-consumption/drinks.csv')

# Visualizando as 6 primeiras linhas dataset
head(data.drinks)
```

```
##             country beer_servings spirit_servings wine_servings
## 1       Afghanistan             0               0             0
## 2           Albania            89             132            54
## 3           Algeria            25               0            14
## 4           Andorra           245             138           312
## 5            Angola           217              57            45
## 6 Antigua & Barbuda           102             128            45
##   total_litres_of_pure_alcohol
## 1                          0.0
## 2                          4.9
## 3                          0.7
## 4                         12.4
## 5                          5.9
## 6                          4.9
```

Análise dos dados

Primeiro temos que explorar e visualizar os dados.

```
# estrutura do dataset drinks.
str(data.drinks)
```

```
## 'data.frame':    193 obs. of  5 variables:
##  $ country                     : Factor w/ 193 levels "Afghanistan",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ beer_servings               : int  0 89 25 245 217 102 193 21 261 279 ...
##  $ spirit_servings             : int  0 132 0 138 57 128 25 179 72 75 ...
##  $ wine_servings               : int  0 54 14 312 45 45 221 11 212 191 ...
##  $ total_litres_of_pure_alcohol: num  0 4.9 0.7 12.4 5.9 4.9 8.3 3.8 10.4 9.7 ...
```

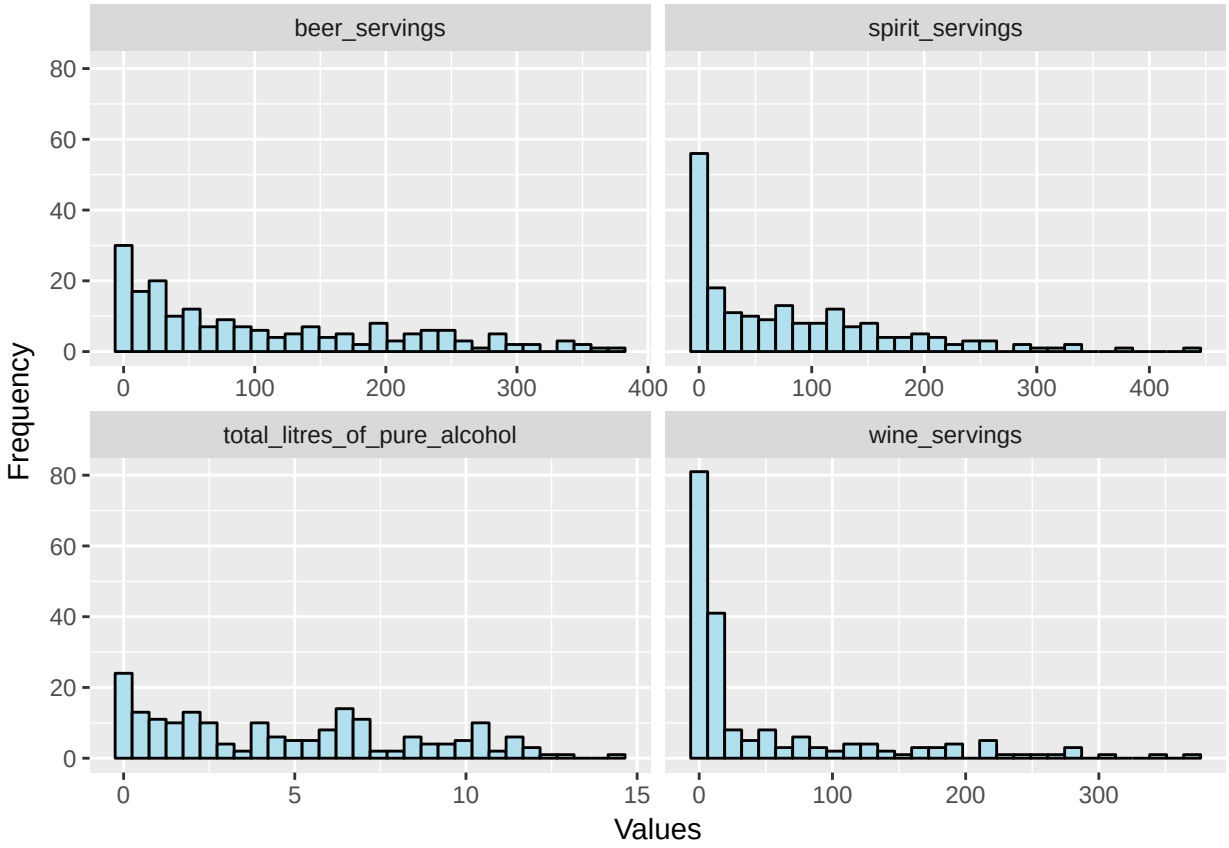
Todas as colunas são expressas como numéricas ou inteiras. E quanto à distribuição estatística?

```
summary(data.drinks)
```

```
##               country    beer_servings   spirit_servings  wine_servings
##  Afghanistan      :  1   Min.   :  0.0   Min.   :  0.00   Min.   :  0.00
##  Albania          :  1   1st Qu.: 20.0   1st Qu.:  4.00   1st Qu.:  1.00
##  Algeria          :  1   Median : 76.0   Median : 56.00   Median :  8.00
##  Andorra          :  1   Mean   :106.2   Mean   : 80.99   Mean   : 49.45
##  Angola           :  1   3rd Qu.:188.0   3rd Qu.:128.00   3rd Qu.: 59.00
##  Antigua & Barbuda:  1   Max.   :376.0   Max.   :438.00   Max.   :370.00
##  (Other)          :187
##  total_litres_of_pure_alcohol
##  Min.   : 0.000
##  1st Qu.: 1.300
##  Median : 4.200
##  Mean   : 4.717
##  3rd Qu.: 7.200
##  Max.   :14.400
##
```

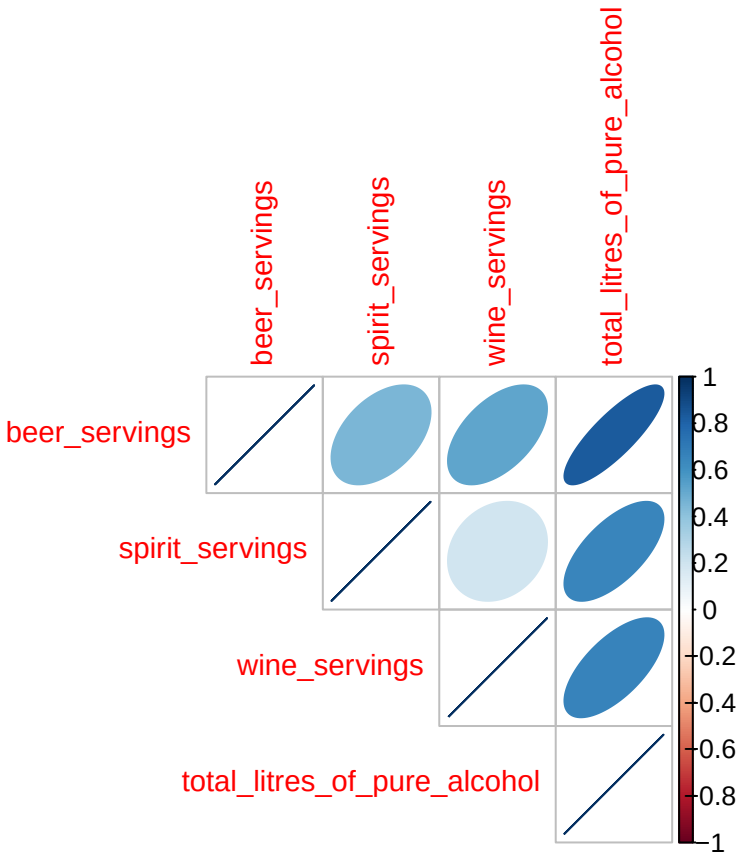
```
#load library
library(tidyverse)
```

```
library(corrplot)
library(gridExtra)
library(GGally)

# Histograma de cada atributo

data.features = data.drinks[,2:5]

data.features %>%
  gather(Attributes, value, 1:4) %>%
  ggplot(aes(x=value)) +
  geom_histogram(fill="lightblue2", colour="black") +
  facet_wrap(~Attributes, scales="free_x") +
  labs(x="Values", y="Frequency")
```



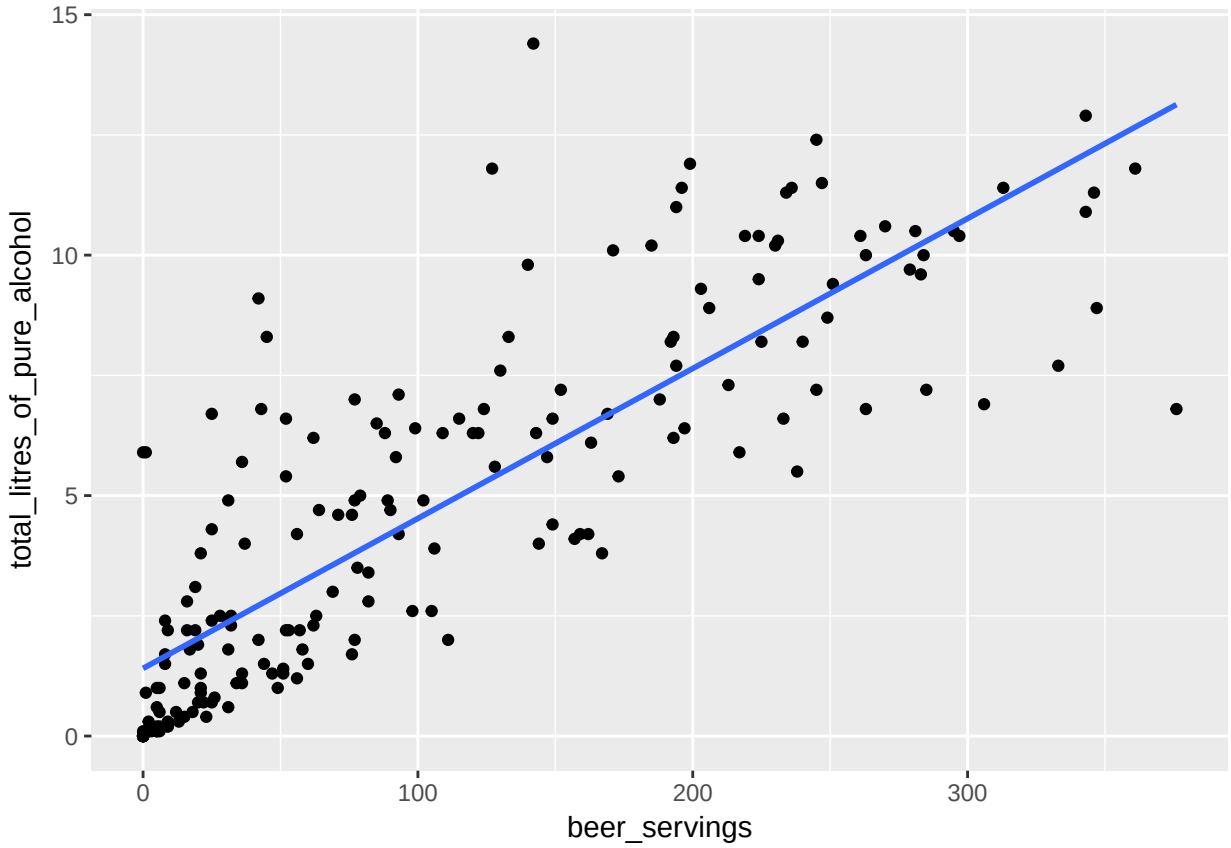
Qual é a relação entre os diferentes atributos? Podemos usar a função **corrplot()** para criar uma exibição gráfica de uma matriz de correlação.

```
# Matrix de correlação
corrplot(cor(data.features), type="upper", method="ellipse", tl.cex=0.9)
```



Existe uma forte correlação linear entre os *beer-services* e os *total-litres-of-pure-alcool*. Podemos modelar a relação entre essas duas variáveis ajustando uma equação linear.

```
# Relationship between beer-services e total-litres-of-pure-alcool.
ggplot(data.features, aes(x=beer_servings, y=total_litres_of_pure_alcohol)) +
  geom_point() +
  geom_smooth(method="lm", se=FALSE)
```

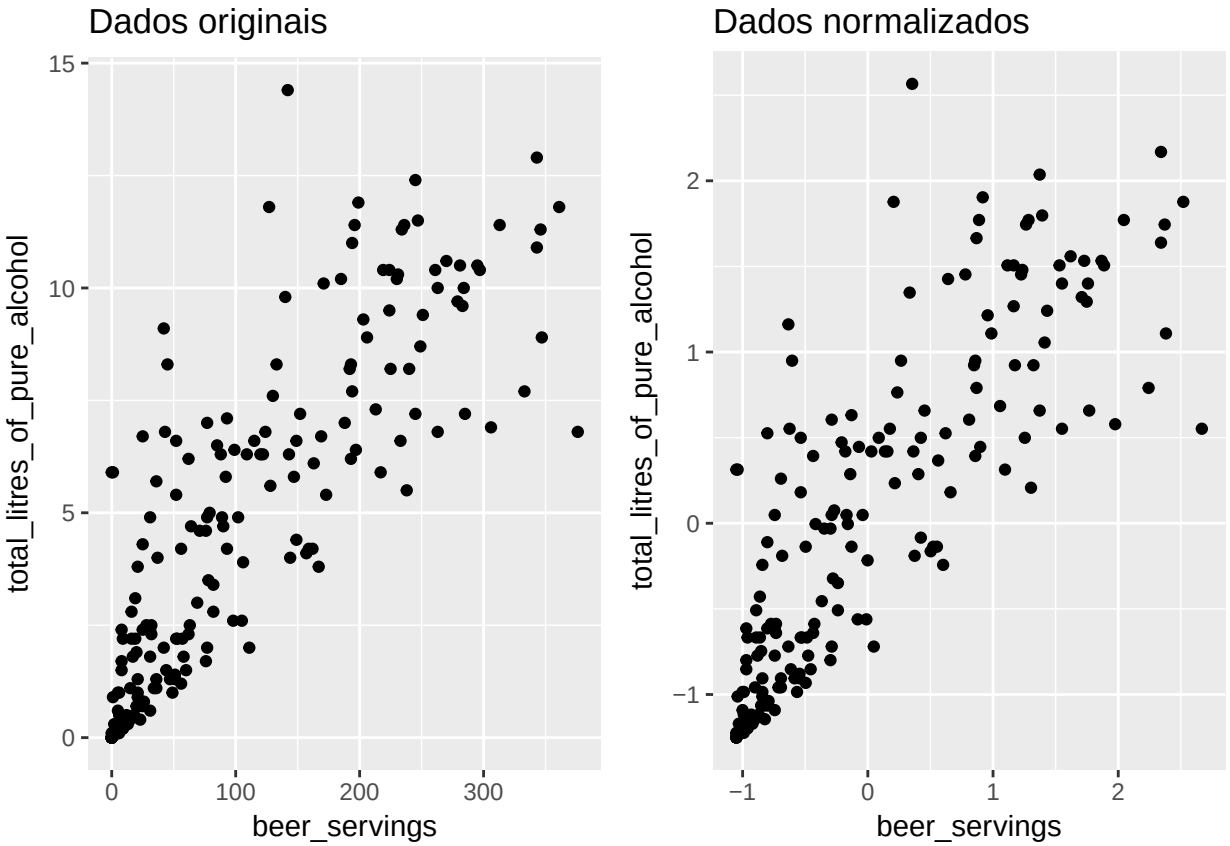


Agora que fizemos uma análise de dados exploratória, podemos preparar os dados para executar o algoritmo k-means.

Preparação dos dados

Temos que normalizar as variáveis para expressá-las no mesmo intervalo de valores. Em outras palavras, normalização significa ajustar os valores medidos em diferentes escalas para uma escala comum.

```
# Normalização
drinks.features.norm <- as.data.frame(scale(data.features))

# dados originais
d1 <- ggplot(data.features, aes(x=beer_servings, y=total_litres_of_pure_alcohol)) +
  geom_point() +
  labs(title="Dados originais")

# dados normalizados
d2 <- ggplot(drinks.features.norm, aes(x=beer_servings, y=total_litres_of_pure_alcohol)) +
  geom_point() +
  labs(title="Dados normalizados")

# subplot
grid.arrange(d1,d2, ncol=2)
```



Os pontos nos dados normalizados são os mesmos que os originais. A única coisa que muda é a escala do eixo.

Execução do k-means

Nesta seção, vamos executar o algoritmo k-means e analisar os principais componentes retornados pela função.

```
# Execução do K-means com k  = 2
set.seed(1234)
drinks.k2 <- kmeans(drinks.features.norm, centers=2)
```

A função kmeans() rertorna um objeto da classe "kmeans" com informações sobre a partição: * cluster. um vetor de inteiros indicando o custer ao qual cada ponto é alocado. * centers. a matriz com o centro dos clusters * size . o número de pontos em cada cluster.

```
# cluster para cada ponto é alocado
drinks.k2$cluster
```

```
##   [1] 2 2 2 1 1 2 1 2 1 1 2 1 2 2 1 1 1 1 2 2 2 2 2 1 2 1 2 2 2 2 2 2 1 2 2
##  [36] 1 2 2 2 2 1 2 1 2 1 1 2 2 1 2 1 1 2 2 2 1 2 1 2 2 1 1 1 2 1 1 2 1 1 2
##  [71] 2 2 1 1 2 1 1 2 2 2 2 1 2 1 2 1 2 1 2 2 2 2 2 1 2 2 2 2 1 1 2 2 2 2 2
## [106] 1 2 2 2 1 2 2 2 2 2 2 2 1 2 2 1 1 2 2 2 1 1 2 2 1 1 2 1 1 2 1 1 2 1 1
## [141] 1 1 2 1 1 1 2 2 2 2 2 1 2 2 2 1 1 2 2 1 1 2 2 1 2 1 1 2 2 1 2 2 2 2 1
## [176] 2 2 2 2 2 1 2 1 2 1 1 2 2 1 2 2 2 2
```

```r
# centro dos clusters
drinks.k2$centers
```

```
##   beer_servings spirit_servings wine_servings total_litres_of_pure_alcohol
## 1     0.9743611       0.7874371     0.7582512                    1.032534
## 2    -0.6329183      -0.5114976    -0.4925393                   -0.670706
```

```r
# tamanho dos cluster
drinks.k2$size
```

```
## [1]  76 117
```

Além disso, a função kmeans () retorna algumas proporções que nos informam quão compacto é um cluster e quão diferentes são os vários clusters entre si. * betweenss - A soma entre os quadrados dos aglomerados. Em uma segmentação ideal, espera-se que essa proporção seja a mais alta possível, já que gostaríamos de ter clusters heterogêneos. * withinss - Vetor da soma de quadrados dentro do cluster, um componente por cluster. Em uma segmentação ideal, espera-se que essa proporção seja a mais baixa possível para cada cluster, uma vez que gostaríamos de ter homogeneidade dentro dos clusters. * tot.withinss - Soma total de quadrados dentro do cluster. * totss - A soma total de quadrados.

```r
# Soma entre os quadrados
drinks.k2$betweenss
```

```
## [1] 402.4935
```

```r
# Soma dos quadrados dentro do cluster
drinks.k2$withinss
```

```
## [1] 264.7697 100.7368
```

```r
# Soma total de quadrados dentro do cluster
drinks.k2$tot.withinss
```

```
## [1] 365.5065
```

```r
# Soma total dos quaddrados
drinks.k2$totss
```

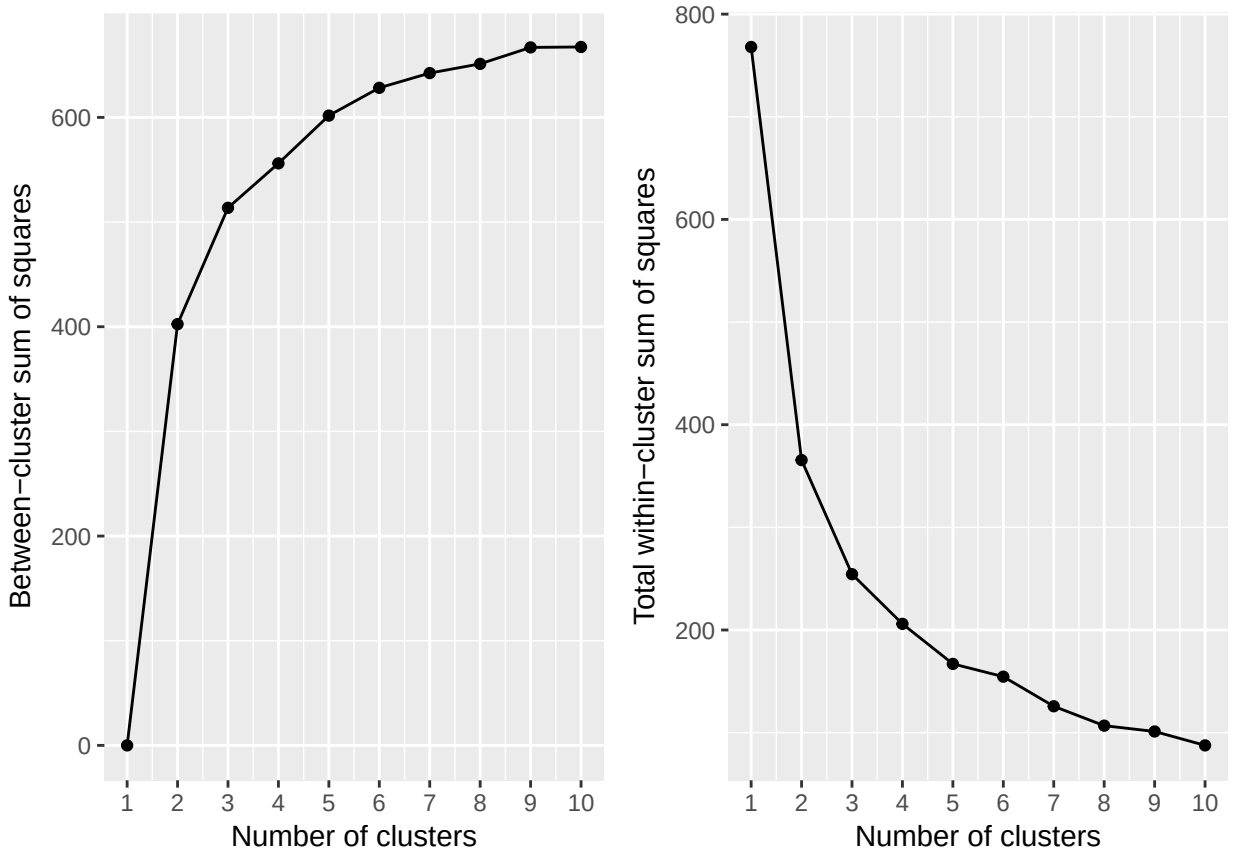
```
## [1] 768
```

Quantos Clustes ?

Para estudar graficamente qual valor de k nos dá a melhor partição, podemos traçar entre e tot.withinss vs Choice de k.

```r
bss <- numeric()
wss <- numeric()

# rodar o algoritmo com diferentes valores de K
set.seed(1234)

for(i in 1:10){

  # para casa k, calcula betweenss e tot.withinss
  bss[i] <- kmeans(drinks.features.norm, centers=i)$betweenss
```

```
  wss[i] <- kmeans(drinks.features.norm, centers=i)$tot.withinss
}

# Soma entre os quadrados dos quadrados vs Escolha de k

d3 <- qplot(1:10, bss, geom=c("point", "line"),
            xlab="Number of clusters", ylab="Between-cluster sum of squares") +
  scale_x_continuous(breaks=seq(0, 10, 1))

# Soma total de quadrados dentro do cluster vs Escolha de k

d4 <- qplot(1:10, wss, geom=c("point", "line"),
            xlab="Number of clusters", ylab="Total within-cluster sum of squares") +
  scale_x_continuous(breaks=seq(0, 10, 1))

# subplot
grid.arrange(d3, d4, ncol=2)
```



Qual é o valor ideal para k? Deve-se escolher um número de clusters para que adicionar outro cluster não forneça uma partição muito melhor dos dados. Em algum momento, o ganho cairá, dando um ângulo no gráfico (critério do cotovelo). O número de clusters é escolhido neste momento. No nosso caso, é claro que 3 é o valor apropriado para k.

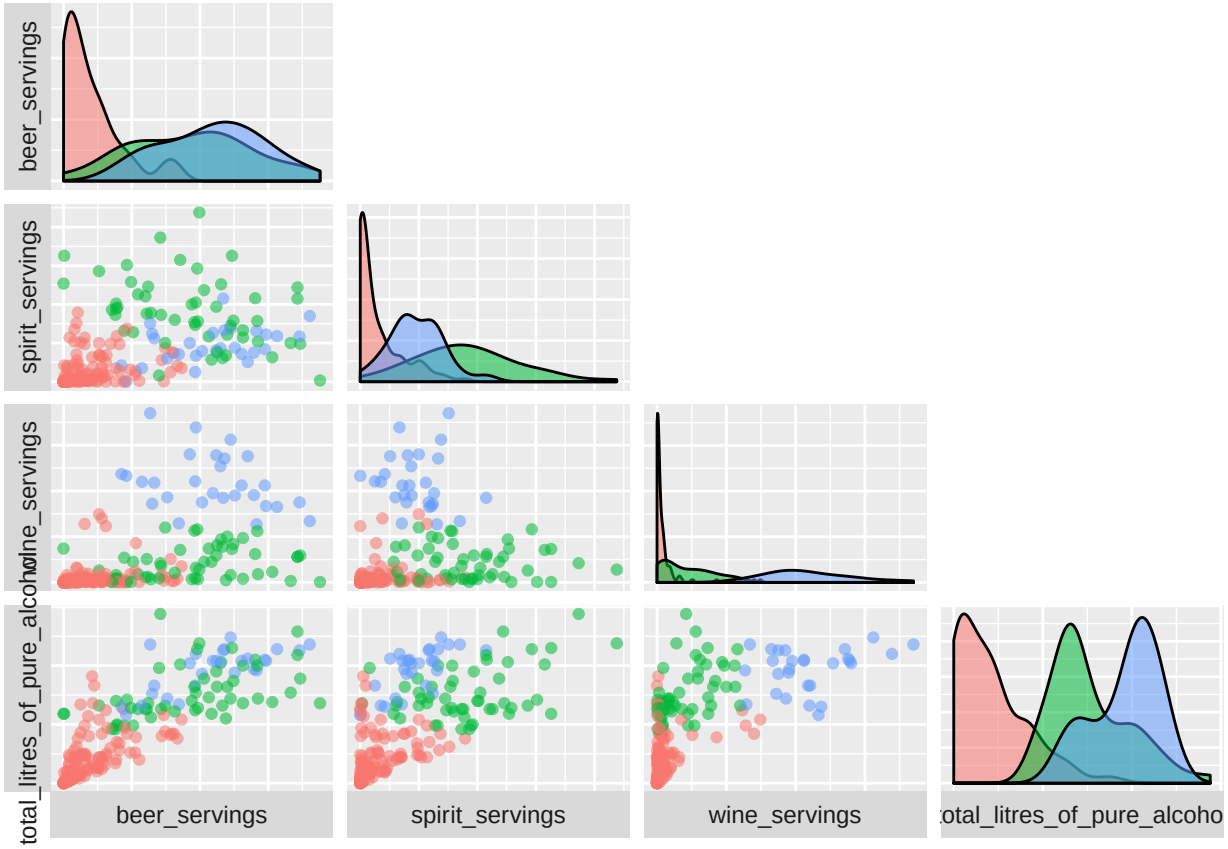
Resultados

```r
# Execução do k-means com k = 3
drinks.k3 <- kmeans(drinks.features.norm, centers = 3)

# Valores médios de cada cluster
aggregate(data.drinks, by=list(drinks.k3$cluster), mean)
```

```
## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical:
## returning NA
```

```
##   Group.1 country beer_servings spirit_servings wine_servings
## 1       1      NA      40.08036        29.40179      10.87500
## 2       2      NA     186.37736       182.35849      41.54717
## 3       3      NA     218.64286        95.50000     218.71429
##   total_litres_of_pure_alcohol
## 1                     2.074107
## 2                     7.690566
## 3                     9.660714
```

```r
# Clustering
ggpairs(cbind(data.features, Cluster=as.factor(drinks.k3$cluster)),
        columns=1:4, aes(colour=Cluster, alpha=0.5),
        lower=list(continuous="points"),
        upper=list(continuous="blank"),
        axisLabels="none", switch="both")
```



Conclusão

Nesta entrada, aprendemos sobre o algoritmo k-means, incluindo a normalização dos dados antes de executá-lo, a escolha do número ideal de clusters (critério de cotovelo) e a visualização do clustering.

Referências

BASE