Lista 6

César A. Galvão - 190011572

Gabriela Carneiro - 180120816

João Vitor Vasconcelos - 170126064

Kevyn Andrade de Souza - 190015853

Índice

Questão 14

Estudar o pacote rsample em https://rsample.tidymodels.org/ e apresentar um exemplo utilizando validação cruzada e Bootstrap.

O pacote "rsample" é um conjunto de funções que tem o intuito gerar diferentes tipos de reamostragem para criar diferentes tipos de amostras e analisar seu desempenho, sendo útil para analisar o desempenho de um modelo com um conjunto de validação.

Bootstrap

O exemplo da utilização do pacote "rsample" para a utilização de bootstrap é feito pela função "bootstraps" em que a função retorna um conjunto de listas em que cada lista contem as informaçoes dos bootstraps realizados. O exemplo foi utilizando o banco de dados "Iris" e realizando o bootstrap 10 vezes.

```
set.seed(123)
boot_splits <- bootstraps(iris, times = 10 )
```

O argumento "Split" retorna uma descrição sobre cada um dos bootstraps realizados em que "analysis" é o conjunto de dados feito por bootstrap, "assessment" são as amostras do banco de original que não foram utilizados no bootstrap e "total" é a quantidade de amostras do banco de dados original que tambem é a mesma quantidade do banco de dados feito por bootstrap.

```
boot_splits$splits
```

```
[[1]]
<Analysis/Assess/Total>
<150/56/150>

[[2]]
<Analysis/Assess/Total>
<150/58/150>

[[3]]
<Analysis/Assess/Total>
<150/50/150>

[[4]]
<Analysis/Assess/Total>
<150/58/150>

[[5]]
<Analysis/Assess/Total>
<150/56/150>

[[6]]
<Analysis/Assess/Total>
<150/55/150>
```

```
[[7]]
<Analysis/Assess/Total>
<150/56/150>

[[8]]
<Analysis/Assess/Total>
<150/52/150>

[[9]]
<Analysis/Assess/Total>
<150/60/150>

[[10]]
<Analysis/Assess/Total>
<150/55/150>
```

Tais partições podem ser chamada pelas funçoes “analysis” e “assessment” respectivamente.

```
boot_splits$splits[[1]] %>%
  analysis() %>%
  head()
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.3	3.0	1.1	0.1	setosa
5.0	3.3	1.4	0.2	setosa
7.7	3.8	6.7	2.2	virginica
4.4	3.2	1.3	0.2	setosa
4.3	3.0	1.1	0.1	setosa
7.7	3.8	6.7	2.2	virginica

```
boot_splits$splits[[1]] %>%
  assessment() %>%
  head()
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
5.0	3.4	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
5.8	4.0	1.2	0.2	setosa
5.1	3.5	1.4	0.3	setosa

A seguir calculamos a média da variável “Sepal.Length” para cada partição de bootstrap feito, podemos ver que os boostraps realizados tem uma média em torno de 5.83, assim podemos confirmar que existe uma consistência entre os bootstraps realizados.

```r
boot_results <- boot_splits %>%
  mutate(mean_sepal_length = map_dbl(splits, ~ {
    train_data <- analysis(.x)
    mean(train_data$Sepal.Length)
  }))

# Visualizar Resultados
boot_results %>%
  dplyr::select(id, mean_sepal_length) %>%
  head()
```

id	mean_sepal_length
Bootstrap01	5.774000
Bootstrap02	5.897333
Bootstrap03	5.924667
Bootstrap04	5.871333
Bootstrap05	5.736667
Bootstrap06	5.793333

Validação cruzada

A validação cruzada é uma técnicas estatítisticas utilizadas para avaliar a adequação de um modelo. Para o exemplo utilizaremos a validação cruzada v-fold(tambem conhecida como k-fold), que consiste em dividir o banco de dado em V subconjuntos, treinando os modelos para V-1 partiçoes e testando o modelo na partição não utilizada. Para exemplificar utilizaremos a função "vfold_cv", em que difente função "bootstraps" a função "analysis" retorna o conjunto de dados utilizados para treino e a função "assessment" retorna o conjunto de dados utilizado para teste.

```r
set.seed(123)
cv_splits <- vfold_cv(iris, v = 5)
cv_splits$splits
```

```
[[1]]
<Analysis/Assess/Total>
<120/30/150>

[[2]]
<Analysis/Assess/Total>
<120/30/150>

[[3]]
<Analysis/Assess/Total>
<120/30/150>

[[4]]
<Analysis/Assess/Total>
<120/30/150>

[[5]]
```

```
<Analysis/Assess/Total>
<120/30/150>
```

Para exemplificar a utilização da validação cruzada, ajustaremos modelos de árvore de decisão (“rpart”) para cada partição da validação cruzada v-fold no seu respectivo conjunto de treino e avaliaremos o quanto o modelo ajustado acerta nas predições feitas no conjunto de teste.

```r
results <- cv_splits %>%
  mutate(model = map(splits, ~ {
    train_data <- analysis(.x)
    test_data  <- assessment(.x)

    model <- train(Species ~ ., data = train_data, method = "rpart")

    predictions <- predict(model, test_data)
    accuracy <- mean(predictions == test_data$Species)

    return(accuracy)
  }))

# Visualizar Resultados
results %>%
  dplyr::select(id, model) %>%
  unnest(model)
```

id	model
Fold1	1.0000000
Fold2	0.8666667
Fold3	0.9000000
Fold4	0.9000000
Fold5	0.9666667

Assim podemos ver que a validação cruzada permite avaliar a performance do modelo de forma robusta, utilizando diferentes subconjuntos dos dados para treino e teste. Isso ajuda a garantir que o modelo se ajusta bem para novos dados.

Questão 15

Selecionar ou gerar um conjunto de dados e comparar a classificação após estimação de densidades utilizando os seguintes métodos:

1. Método do Histograma
2. Estimação baseada em Núcleos
3. k-Vizinhos mais Próximos

Como nas notas de aula o exemplo usado para a classificação foi o conjunto de dados `faithful`, vamos usar o conjunto `iris` para comparar a classificação após estimação de densidades utilizando os métodos citados.

Método do Histograma

Com o método do histograma, pretende-se agregar $\mathbf{X} = \{X_1, \ldots, X_n\}$, $X_i \overset{iid}{\sim} f(x)$ em intervalos da forma $[x_0, x_0 + h)$ e usar a frequência relativa de $\{x_n\}$ para aproximar a densidade $f(x)$. A estimativa é feita por

$$f(x_0) = F'(x_0) = \lim_{h \to 0^+} \frac{F(x_0 + h) - F(x_0)}{h} = \lim_{h \to 0^+} \frac{P[x_0 < X < x_0 + h]}{h}, \tag{1}$$

estabelecida a partir de uma origem t_0 e um comprimento (*binwidth*) $h > 0$. O histograma é então construído contando o número de pontos em intervalos chamados de *bins*, definidos por $\{I_k : [t_k, t_{k+1}); \, t_k = t_0 + hk, \, k \in \mathcal{Z}\}$. O histograma de densidade no ponto x é definido como

$$\hat{f}_H(x; t_0, h) = \frac{1}{nh} \sum_{i=1}^{n} \mathbf{1}_{\{x_i \in I_k\}}.$$

Fica evidente a dependência em t_0, que pode ser evitada com um estimador *naïve* dado por

$$f(x) = F'(x) = \lim_{h \to 0^+} \frac{F(x + h) - F(x - h)}{2h} = \lim_{h \to 0^+} \frac{P[x - h < X < x + h]}{2h}.$$

Dessa forma, o histograma no ponto x é definido como

$$\hat{f}_N(x; h) = \frac{1}{2nh} \sum_{i=1}^{n} \mathbf{1}_{\{x - h < X_i < x + h\}}.$$

Para comparação, vamos usar o pacote `ggplot2` para gerar o histograma da variável `Petal.Length` do conjunto de dados `iris`, considerando $h = 0,2$.

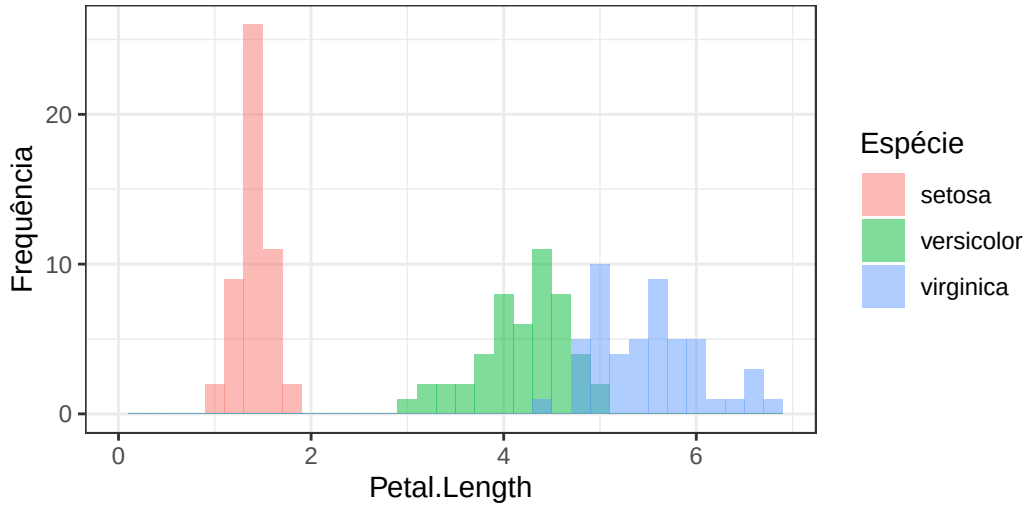


Figura 1: Histograma de Petal Length do conjunto de dados iris.

Os histogramas na Figura 2 foram criados manualmente com o mesmo binwidth, porém com origens variando entre $t_0 = 0$ e $t_0 = 1$ para ilustrar a limitação da dependência deste termo. Depreende-se portanto que está sendo usada a equação 1.

```
# bindiwdth
bk1 <- seq(0.5,7.1,by = 0.2)
bk2 <- seq(1, 7, by = 0.2)

par(mfrow = c(1,2))

hist(iris$Petal.Length, probability = T, breaks = bk1,
     ylim = c(0,0.8), col = 'blue',
     xlab = 't0 = 0.5, h = 0.2', main = "")
rug(iris$Petal.Length)

hist(iris$Petal.Length, probability = T, breaks = bk2,
     ylim = c(0,0.8), col = 'blue',
     xlab = 't0 = 1, h = 0.2', main = "")
rug(iris$Petal.Length)
```

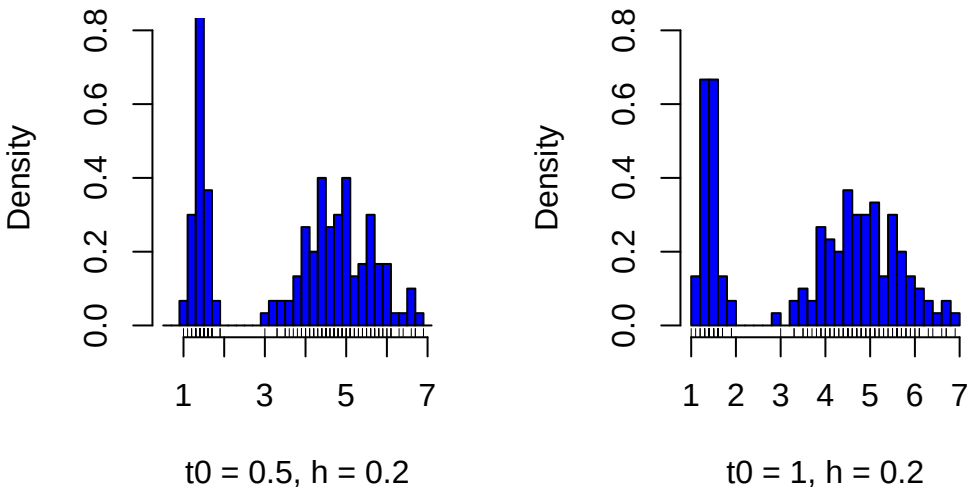


Figura 2: Histograma manual de Petal Length do conjunto de dados iris utilizando h = 0,2.

Para a classificação, será determinada uma fronteira entre as espécies. Além disso, serão reservados pontos para treino e alguns para teste, que serão usados nos demais itens desta questão.

```
# iris com as variáveis selecionadas
iris2 <- iris %>% dplyr::select(Petal.Length, Petal.Width, Species)

# separação arbitrária de treino e teste
iris2 <- iris2 %>%
  mutate(partition = case_when(
    Petal.Length == 5.1 & Petal.Width %in% c(1.5, 1.6) ~ "teste",
    Petal.Length == 5 & Petal.Width %in% c(1.7, 1.5) ~ "teste",
    Petal.Length == 1.9 & Petal.Width == 0.4 ~ "teste",
    TRUE ~ "treino"
  ))
```

Os histogramas para cada espécie são estimados a seguir:

```
#setosa
hist_setosa <- iris2 %>%
  dplyr::filter(Species == "setosa", partition == "treino") %$%
  hist(Petal.Length, probability = T, breaks = bk1)
```

```
#virginica
hist_virginica <- iris2 %>%
  dplyr::filter(Species == "virginica", partition == "treino") %$%
  hist(Petal.Length, probability = T, breaks = bk1)
```

```r
#versicolor
hist_versicolor <- iris2 %>%
  dplyr::filter(Species == "versicolor", partition == "treino") %$%
  hist(Petal.Length, probability = T, breaks = bk1)
```

Finalmente, são acessados os resultados dos histogramas para construir a Tabela 5 a seguir, em que apenas as *bins* com contagens superiores a zero são exibidas. As colunas da tabela correspondem ao valor inicial de cada *bin*, a contagem de observações, a densidade e a espécie.

Pode-se observar que a espécie Setosa está em uma região de comprimento de pétala bem separada das demais, então, para a classificação, basta verificar se uma nova observação se encontra na mesma região que as demais. Para as espécies Virginica e Versicolor, a separação não é tão clara, então é necessário adotar uma regra. Como a densidade do *bin* com início em 4.5 é maior para Versicolor e o inverso ocorre para o *bin* com início em 4.7, a fronteira entre esses bins será a regra de decisão. Isso implica em prováveis 2% de erro de classificação das Virgínica, quando se classifica em Versicolor, e 8.4% de erro de classificação das Versicolor, quando se classifica em Virginica[1].

```r
options(knitr.kable.NA = '-')

#monta a tabela com dados dos histogramas
tibble(
  ti = hist_setosa$breaks[-34],
  cont.setosa = hist_setosa$counts,
  dens.setosa = hist_setosa$density,
  cont.virginica = hist_virginica$counts,
  dens.virginica = hist_virginica$density,
  cont.versicolor = hist_versicolor$counts,
  dens.versicolor = hist_versicolor$density
) %>%
  # seleciona linhas com valores maiores que zero
  dplyr::filter(if_any(-ti, ~ . > 0)) %>%
  # ajusta decimais para arrumar o tamanho da tabela
  mutate(across(everything(), ~ round(., 2)),
         across(everything(), ~ if_else(. == 0, NA, .))) %>%
  # gera tabela tex
  knitr::kable(
    align = "c"
  )
```

Tabela 5: Histogramas de Petal Length do conjunto de dados iris.

ti	cont.setosa	dens.setosa	cont.virginica	dens.virginica	cont.versicolor	dens.versicolor
0.9	2	0.20	-	-	-	-
1.1	9	0.92	-	-	-	-
1.3	26	2.65	-	-	-	-
1.5	11	1.12	-	-	-	-

[1]Erros calculados a partir da sobreposição dos histogramas, considerando a fronteira de decisão.

Tabela 5: Histogramas de Petal Length do conjunto de dados iris.

ti	cont.setosa	dens.setosa	cont.virginica	dens.virginica	cont.versicolor	dens.versicolor
1.7	1	0.10	-	-	-	-
2.9	-	-	-	-	1	0.10
3.1	-	-	-	-	2	0.21
3.3	-	-	-	-	2	0.21
3.5	-	-	-	-	2	0.21
3.7	-	-	-	-	4	0.42
3.9	-	-	-	-	8	0.83
4.1	-	-	-	-	6	0.62
4.3	-	-	1	0.10	11	1.15
4.5	-	-	-	-	8	0.83
4.7	-	-	5	0.52	4	0.42
4.9	-	-	8	0.83	-	-
5.1	-	-	4	0.42	-	-
5.3	-	-	5	0.52	-	-
5.5	-	-	9	0.94	-	-
5.7	-	-	5	0.52	-	-
5.9	-	-	5	0.52	-	-
6.1	-	-	1	0.10	-	-
6.3	-	-	1	0.10	-	-
6.5	-	-	3	0.31	-	-
6.7	-	-	1	0.10	-	-

Em posse das regras de decisão, classifica-se os pontos de teste e se obtém a Tabela 6 a seguir.

```
iris2 %>%
  dplyr::filter(partition == "teste") %>%
  dplyr::select(-Petal.Width, -partition) %>%
  mutate(`Classificação` = c("setosa", rep("virginica", 4))) %>%
  knitr::kable(align = "cll")
```

Tabela 6: Classificação dos pontos de teste do conjunto de dados iris via histograma.

Petal.Length	Species	Classificação
1.9	setosa	setosa
5.0	versicolor	virginica
5.1	versicolor	virginica
5.0	virginica	virginica
5.1	virginica	virginica

Estimação baseada em Núcleos

Soliciona outro problema do Método do Histograma, que é a necessidade de se utilizar intervalos pequenos e n grande para se obter uma boa aproximação da densidade alvo. O estimador de densidade no caso univariado é dado por

$$\hat{f}(x;h) = \frac{1}{n}\sum_{i=1}^{n} k\, h(x - X_i),$$

enquanto no caso multivariado é dado por

$$\hat{f}(x;h) = \frac{1}{n|\mathbf{H}|^{1/2}}\sum_{i=1}^{n} k(\mathbf{H}^{-1/2}(\mathbf{x} - \mathbf{X}_i)).$$

Nas expressões k é o núcleo de uma função densidade arbitrária. Um problema que surge é a seleção do bandwidth h, que será demonstrado a seguir utilizando o pacote `ks` para a obtenção da matriz $\mathbf{H}_{p\times p}$ de intervalos[2].

A seguir é estimada a densidade conjunta para as variáveis `Petal.Length` e `Petal.Width` com o conjunto de teste, seguida da Figura 3 ilustrando-a. Os pontos de cor azul correspondem aos pontos de teste.

```
# obtenção da matriz H
He <- iris2 %>%
  dplyr::filter(partition == "treino") %>%
  dplyr::select(Petal.Length, Petal.Width) %>%
  ks::Hpi()

# kernel density estimation com a H estimada
kdeHe <- iris2 %>%
  dplyr::filter(partition == "treino") %>%
  dplyr::select(Petal.Length, Petal.Width) %>%
  ks::kde(., H=He)
```

```
pteste <- iris2 %>%
  dplyr::filter(partition == "teste") %>%
  dplyr::select(Petal.Length, Petal.Width) %>%
  as.matrix()

par(mar = c(3,4,4,4), pin = c(4, 2.75))

image(kdeHe$eval.points[[1]],kdeHe$eval.points[[2]],
      kdeHe$estimate, xlab = 'Petal Length',
      ylab = 'Petal Width')
```

[2]Para o caso unidimensional, i.e. $p = 1$, $\mathbf{H} = h^2$.

```
points(kdeHe$x, col = alpha("black", 0.3))
points(pteste, col = "blue", pch = 8)
```

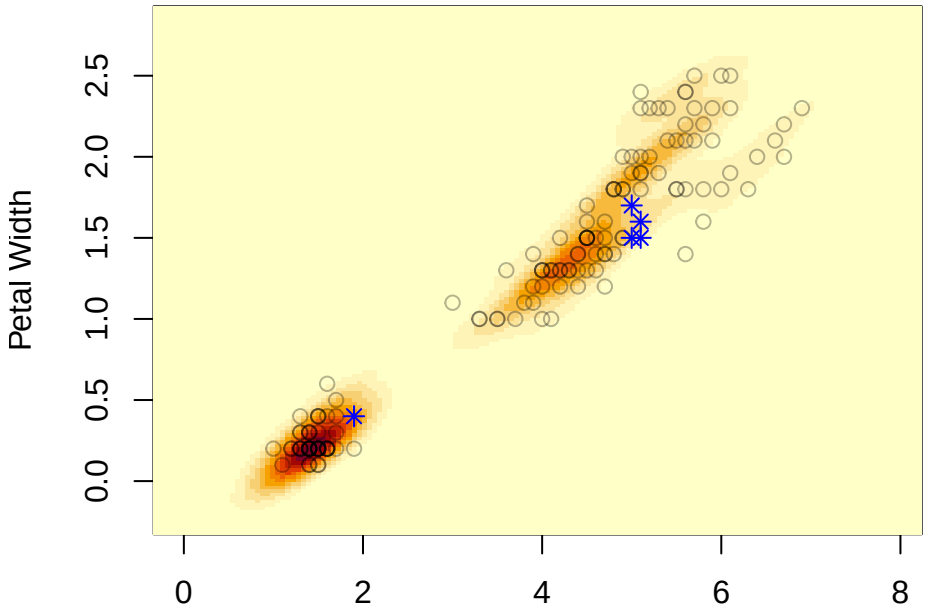


Figura 3: Densidade estimada do conjunto de dados de treino iris para comprimento e largura da pétala.

Como a classificação da espécia Setosa é novamente óbvia, apenas a classificação para as demais espécies será feita. As densidades para Virgínica e Versicolor são estimadas a seguir.

```
# H versicolor
He_versi <- iris2 %>%
  dplyr::filter(partition == "treino", Species == "versicolor") %>%
  dplyr::select(Petal.Length, Petal.Width) %>%
  ks::Hpi()

# kde versi
kdeHe_versi <- iris2 %>%
  dplyr::filter(partition == "treino", Species == "versicolor") %>%
  dplyr::select(Petal.Length, Petal.Width) %>%
  ks::kde(., H=He_versi)

# H virginica
He_virg <- iris2 %>%
  dplyr::filter(partition == "treino", Species == "virginica") %>%
  dplyr::select(Petal.Length, Petal.Width) %>%
  ks::Hpi()

# kde virginica
```

```r
kdeHe_virg <- iris2 %>%
  dplyr::filter(partition == "treino", Species == "virginica") %>%
  dplyr::select(Petal.Length, Petal.Width) %>%
  ks::kde(., H=He_virg)
```

Os valores das densidades para os pontos de teste são dados na Tabela 7 a seguir, assim como as suas classificações.

```r
pteste2 <- iris2 %>%
  dplyr::filter(partition == "teste", Species != "setosa") %>%
  dplyr::select(Petal.Length, Petal.Width) %>%
  as.matrix()

iris2 %>%
  dplyr::filter(partition == "teste", Species != "setosa") %>%
  dplyr::select(Petal.Length, Petal.Width, Species) %>%
  mutate(dens.versi = predict(kdeHe_versi, x = pteste2),
         dens.virg = predict(kdeHe_virg, x = pteste2),
         `Classificação` = if_else(dens.versi > dens.virg, "versicolor",
         ↪  "virginica")) %>%
  knitr::kable(align = "cclccl",
               digits = 2)
```

Tabela 7: Classificação dos pontos de teste com densidades estimadas para Virgínica e Versicolor.

Petal.Length	Petal.Width	Species	dens.versi	dens.virg	Classificação
5.0	1.7	versicolor	0.27	0.46	virginica
5.1	1.6	versicolor	0.45	0.15	versicolor
5.0	1.5	virginica	0.84	0.04	versicolor
5.1	1.5	virginica	0.42	0.04	versicolor

k-Vizinhos mais Próximos (KNN)

O método de k-vizinhos mais próximos (*k nearest neighbors*) pretende classificar um determinado ponto $\mathbf{x}_0$ a partir dos k pontos mais próximos. A classificação é dada em função da menor distância euclidiana entre $\mathbf{x}_0$ e os pontos de treinamento e, se houver empate, a escolha é feita aleatoriamente. Enquanto há muitas medidas de distância possíveis, a mais comum é a distância euclidiana, dada por

$$d(\mathbf{x}_i, \mathbf{x}_0) = \sqrt{\sum_{l=1}^{p}(x_{il} - x_{0l})^2} = ||\mathbf{x}_i - \mathbf{x}_0||_2.$$

A Figura 4 apresenta um gráfico de dispersão do conjunto de dados em função das variáveis de interesse, no qual as cores indicam a espécie e as formas indicam a partição de treino ou teste. As unidades de teste foram selecionadas de forma arbitrária visando ilustrar as limitações desse método.

Enquanto é esperado que o ponto de teste de espécie Setosa seja classificado corretamente, não se pode esperar uma classificação correta dos pontos de teste de espécie Versicolor e Virginica, visto que estão numa fronteira difusa entre os grupos.

```
# gráfico
ggplot() +
  geom_point(data = iris2[iris2$partition == "teste",], aes(x = Petal.Length, y =
  ↪  Petal.Width, color = Species), shape = 8) +
  geom_point(data = iris2[iris2$partition == "treino",], aes(x = Petal.Length, y =
  ↪  Petal.Width, color = Species), shape = 16) +
  theme_bw() +
  labs(x = "Petal.Length",
    y = "Petal.Width",
    color = "Espécie",
    shape = "Partição")
```

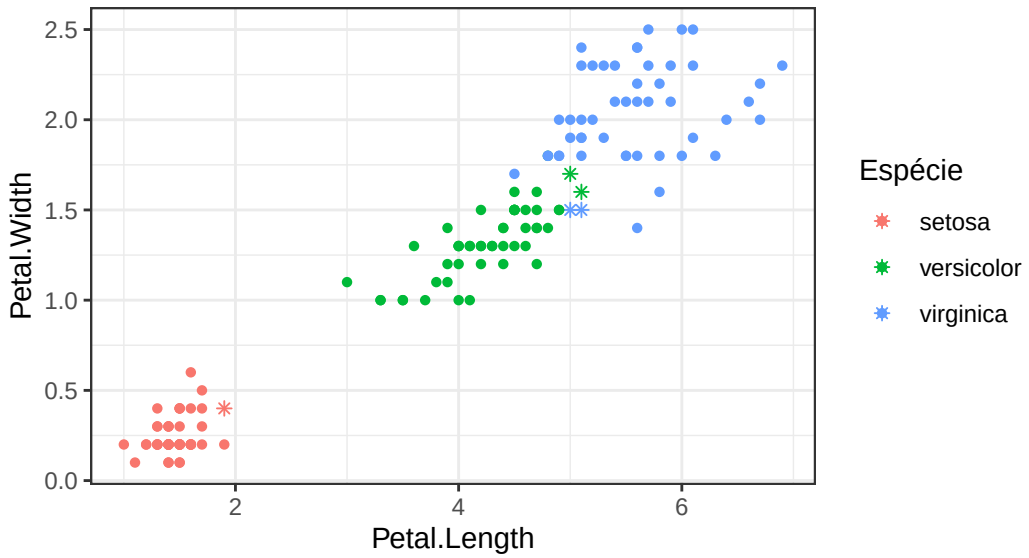


Figura 4: Gráfico de dispersão do conjunto de dados iris.

A Tabela 8 apresenta a espécie real dos pontos de teste, bem como a classificação utilizando os 3 e cinco vizinhos mais próximos (knn = 3 e knn = 5 respectivamente). O comportamento é exatamente o esperado.

```
treino <- iris2 %>% filter(partition == "treino")
teste <- iris2 %>% filter(partition == "teste")
```

```r
knn3 <- knn(train = treino[, c(1,2)],
    test = teste[, c(1,2)],
    cl = treino$Species,
    k = 3)

knn5 <- knn(train = treino[, c(1,2)],
    test = teste[, c(1,2)],
    cl = treino$Species,
    k = 5)

data.frame(
  teste$Species,
  knn3,
  knn5
) %>%
  knitr::kable(
    col.names = c("Espécie real", "k = 3", "k = 5")
  )
```

Tabela 8: Classificação dos pontos de teste do conjunto de dados iris.

Espécie real	k = 3	k = 5
setosa	setosa	setosa
versicolor	virginica	virginica
versicolor	versicolor	virginica
virginica	versicolor	versicolor
virginica	versicolor	virginica

Finalmente, projeta-se a classificação dos pontos de acordo com os dois algoritmos KNN utilizados na Figura 5. O conjunto completo é exibido no gráfico central da segunda linha.

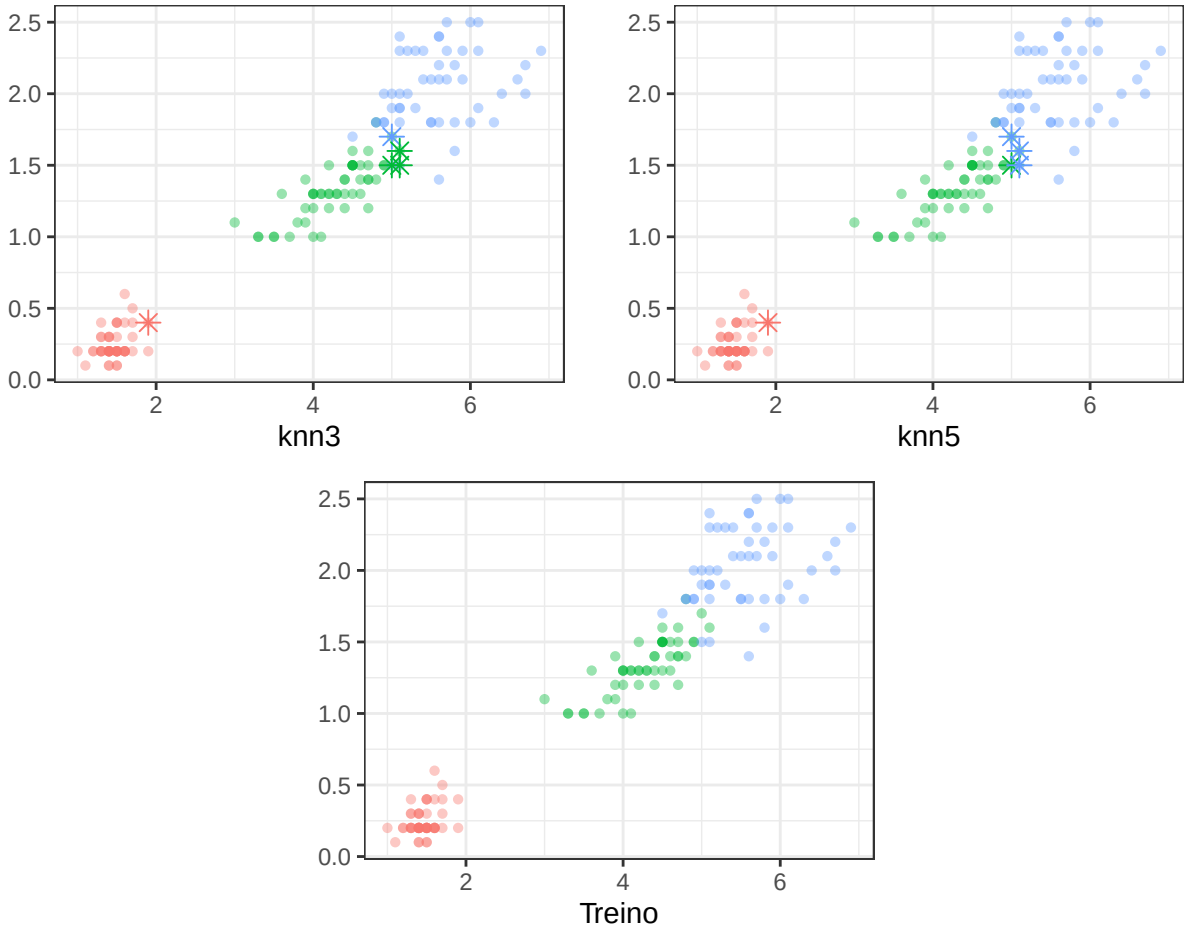


Figura 5: Classificação dos pontos de teste do conjunto de dados iris.

Questão 16

Estudar o pacote `ks` do R e apresentar um exemplo.

Pacote KS

O pacote `ks` é usado para análise de suavização de kernel e estimativa de densidade de kernel (KDE). Ele oferece ferramentas não paramétricas para estimar a densidade de uma ou mais variáveis e suavizar dados.

Existem três tipos principais de funções neste pacote:

1. **Cálculo de estimadores de kernel** - os nomes dessas funções começam com 'k'.

2. **Cálculo de seletores de bandwidth** - começam h para dados unidimensionais ou H para dados multivariados.
3. **Exibição de estimadores de kernel** - começam com 'plot'.

O kernel usado em todo o pacote é o kernel normal (Gaussiano) K. Para dados unidimensionais, o bandwidth h é o desvio padrão do kernel normal, enquanto para dados multivariados, a matriz de bandwidths H é a matriz de variância.

Função `kde`

A KDE é uma técnica específica de Kernel Smoothing usada para estimar a função densidade de probabilidade de uma variável aleatória. Dessa forma, para um conjunto de dados $X = (x_1, x_2, \ldots, x_n)$:

$$\hat{f}(\mathbf{x}) = n^{-1} \sum_{i=1}^{n} K_H(\mathbf{x} - \mathbf{X}_i), \tag{2}$$

onde:

- $\hat{f}(x)$ é a densidade estimada no ponto x.
- n é o número de pontos de dados.
- H é a matriz de bandwidth.
- K é a função kernel.

Exemplo prático

Para o exemplo prático, vamos utilizar o bando de dados *Iris*.

```
# Carregando o conjunto de dados iris
data(iris)
```

Para a análise univariada de cada uma das variáveis presentes no conjunto de dados, temos:

```
# Função para calcular a densidade e criar um data frame
calcular_densidade <- function(x) {
  est_densidade <- ks::kde(x = x)
  data.frame(x = est_densidade$eval.points, y = est_densidade$estimate)
}

# Calculando a densidade para cada variável
densidade_sepal_length <- calcular_densidade(iris$Sepal.Length)
densidade_sepal_width <- calcular_densidade(iris$Sepal.Width)
densidade_petal_length <- calcular_densidade(iris$Petal.Length)
densidade_petal_width <- calcular_densidade(iris$Petal.Width)
```

```r
# Adicionando a variável correspondente ao data frame
densidade_sepal_length$Variable <- "Sepal.Length"
densidade_sepal_width$Variable <- "Sepal.Width"
densidade_petal_length$Variable <- "Petal.Length"
densidade_petal_width$Variable <- "Petal.Width"

# Combinando todos os data frames
densidade_total <- rbind(densidade_sepal_length, densidade_sepal_width,
↪  densidade_petal_length, densidade_petal_width)
```

```r
ggplot(densidade_total, aes(x = x, y = y, color = Variable)) +
  geom_line() +
  labs(x = "Valor da Variável",
       y = "Densidade") +
  theme_bw() +
  theme(legend.title = element_blank(),
        panel.border = element_blank(),
        axis.line = element_line())
```

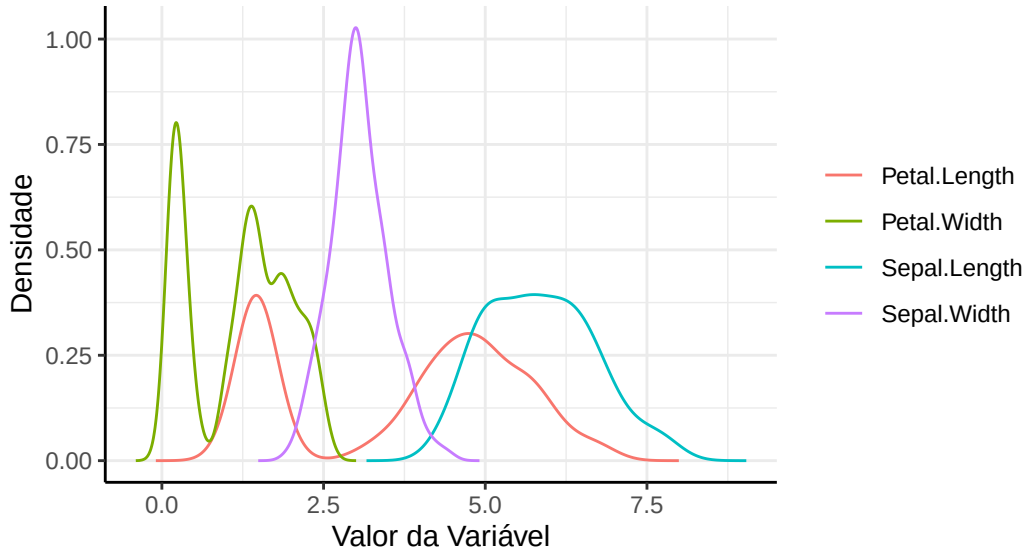


Figura 6: Estimativa de densidade kernel para as variáveis do conjunto de dados iris.

A Figura 6 exibe as estimativas de densidade kernel para todas as variáveis do conjunto de dados iris: `Sepal.Length`, `Sepal.Width`, `Petal.Length` e `Petal.Width`. Cada variável é representada por uma linha de cor diferente, facilitando a comparação visual das distribuições dessas variáveis.

A linha azul, representando o comprimento da sépala, mostra um pico significativo em torno de 5 a 6 cm, indicando que muitas observações possuem comprimentos de sépala nessa faixa. A linha roxa, que representa a largura da sépala, apresenta um pico principal em torno de 3 cm, indicando uma alta concentração de observações nesse valor.

Para os dados de pétala, a linha vermelha, que representa o comprimento das pétalas, possui dois picos

principais: um em torno de 1,5 cm e outro próximo a 5 cm, sugerindo duas distribuições distintas, possivelmente correspondentes a diferentes espécies. A largura da pétala, representada pela linha verde, apresenta um pico significativo próximo a 0,2 cm e um segundo pico em torno de 1,5 cm, também possivelmente correspondendo a duas espécies diferentes.

Assim, as diferentes linhas de densidade mostram onde os valores dessas variáveis se concentram, ajudando a identificar padrões e diferenças entre elas.

Para a análise multivariada, utilizaremos os dados de pétala em uma análise e os dados de sépala em outra.

Para os dados de sépala, temos:

```
# Selecionando duas colunas para análise multivariada para dados de Sépala
sep<- as.matrix(iris[, 1:2])

# Estimando a densidade usando kernel para dados multivariados
est_densidade_multivariada_sep <- kde(x = sep, compute.cont = TRUE)

# Obtendo os pontos de avaliação e a densidade estimada
eval_points_sep <- expand.grid(est_densidade_multivariada_sep$eval.points[[1]],
↪  est_densidade_multivariada_sep$eval.points[[2]])
density_sep <- as.vector(est_densidade_multivariada_sep$estimate)
densidade_df_sep <- data.frame(eval_points_sep, density_sep)

# Renomeando colunas
colnames(densidade_df_sep) <- c("Sepal.Length", "Sepal.Width", "Density")
```

```
# Visualizando a densidade estimada para duas dimensões com GGPLOT
ggplot(densidade_df_sep, aes(x = Sepal.Length, y = Sepal.Width, fill = Density)) +
  geom_tile() +
  labs(x = "Sepal Length",
       y = "Sepal Width") +
  scale_fill_gradient(low = "white", high = "blue") +
  theme_bw()+
  theme(panel.grid = element_blank())
```

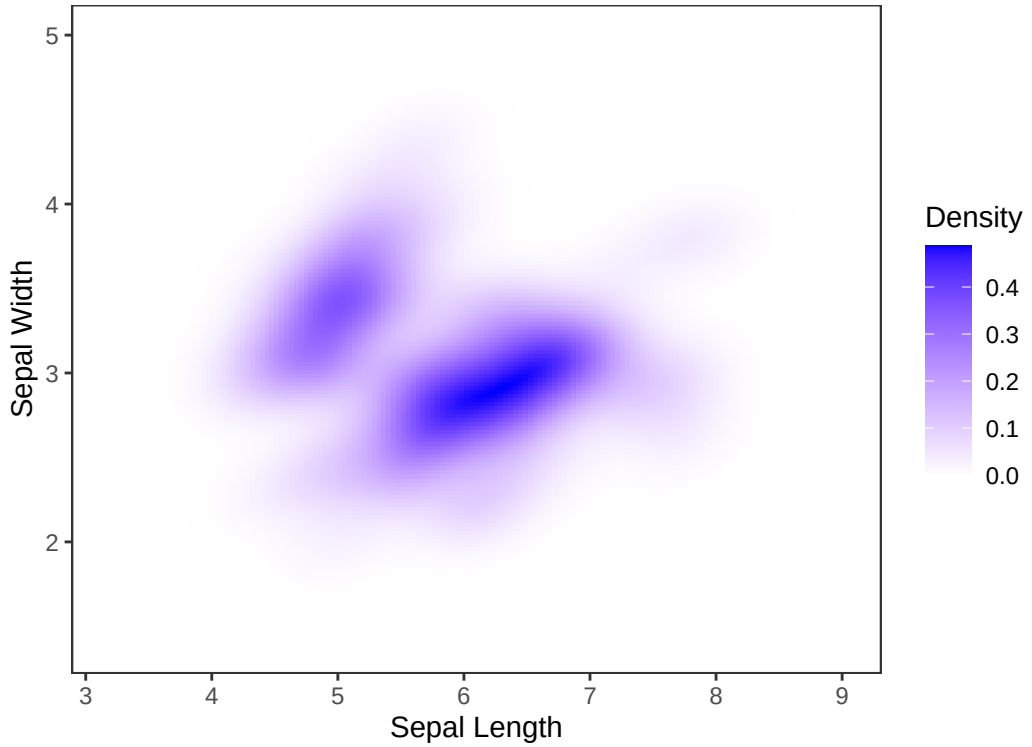


Figura 7: Estimativa de densidade kernel multivariada para as variáveis de sépala do conjunto de dados iris.

A Figura 7 mostra a estimativa de densidade kernel bivariada para as variáveis: `Sepal.Length` e `Sepal.Width`. No eixo X, temos os valores do comprimento da sépala, enquanto no eixo Y estão os valores da largura da sépala. A densidade é representada por um mapa de calor, onde a intensidade da cor azul indica a magnitude da densidade, variando de claro (baixa densidade) a escuro (alta densidade).

As áreas de cor azul mais escura indicam onde há maior concentração de observações, revelando duas regiões principais de alta densidade. As áreas de cor azul mais clara, por outro lado, indicam menor concentração de observações, sugerindo menos ocorrência de pares de valores nessas regiões.

A presença de duas regiões de alta densidade sugere uma distribuição multimodal, indicando a existência de diferentes grupos ou espécies dentro do conjunto de dados. Já a dispersão dos valores indica variabilidade nas medidas de comprimento e de lagura da sépala.

Já para os dados de pétala temos:

```
### Pétala
# Selecionando duas colunas para análise multivariada para dados de Sépala
pet <- as.matrix(iris[, 3:4])

# Estimando a densidade usando kernel para dados multivariados
est_densidade_multivariada_pet <- kde(x = pet, compute.cont = TRUE)
```

```r
# Obtendo os pontos de avaliação e a densidade estimada
eval_points_pet <- expand.grid(est_densidade_multivariada_pet$eval.points[[1]],
↪  est_densidade_multivariada_pet$eval.points[[2]])
density_pet <- as.vector(est_densidade_multivariada_pet$estimate)
densidade_df_pet <- data.frame(eval_points_pet, density_pet)

# Renomeando colunas
colnames(densidade_df_pet) <- c("Petal.Length", "Petal.Width", "Density")
```

```r
# Visualizando a densidade estimada para duas dimensões com GGPLOT
ggplot(densidade_df_pet, aes(x = Petal.Length, y = Petal.Width, fill = Density)) +
  geom_tile() +
  labs( x = "Petal Length",
        y = "Petal Width") +
  scale_fill_gradient(low = "white", high = "blue") +
  theme_bw()+
  theme(panel.grid = element_blank())
```

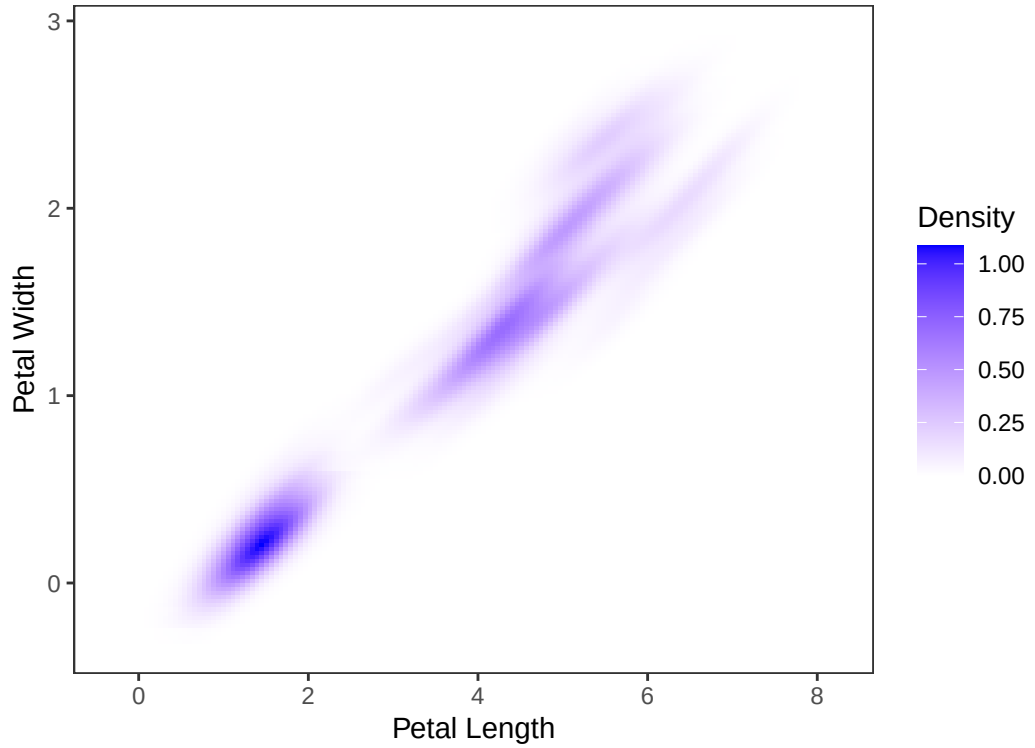


Figura 8: Estimativa de densidade kernel multivariada para as variáveis de pétala do conjunto de dados iris.

A Figura 8 mostra a estimativa de densidade kernel bivariada para as variáveis `Petal.Length` e `Petal.Width`. No eixo X, estão os valores do comprimento da pétala, enquanto no eixo Y estão

os valores da largura da pétala.

A dispersão da densidade ao longo do gráfico sugere uma relação linear entre largura e comprimento da pétala, indicando que à medida que um característica aumenta, a outra também tende a aumentar.

Questão 17

Apresentar um exemplo com classificador LDA e QDA.

Como auxiliar, será definido um método `is_cov`, para garantir que as matrizes de variância-covariância (que serão geradas aleatoriamente) satisfazem as condições necessárias. Será definida também uma função que calcula a acurácia de um modelo.

```
is_cov <- function(Sigma) {

  is_square <- nrow(Sigma) == ncol(Sigma)

  is_symmetric <- all(t(Sigma) == Sigma)

  positive_diag <- all(diag(Sigma) > 0)

  eigen_greater_zero <- all(with(eigen(Sigma), values) > 0)

  cov_smaller_than_sds <- TRUE

  for (row in 1:p) {
    for (col in row:p) {
      cov_smaller_than_sds <- cov_smaller_than_sds & (
        abs(Sigma[row, col]) <= sqrt(Sigma[row, row] * Sigma[col, col])
      )
    }
  }

  all(
    is_square, is_symmetric, positive_diag,
    cov_smaller_than_sds, eigen_greater_zero
  )
}

accuracy <- function(model, data, response) {
  prediction <- predict(model, data)$class

  round(sum(prediction == response) / length(response), 3)
}
```

Em seguida, será criada uma matriz de variâncias-covariâncias, a partir da qual será criada uma normal multivariada. Essa matriz será multiplicada pela sua transposta, de modo que satisfaça as condições necessárias para ser uma matriz de variância-covariância.

```r
p <- 3

set.seed(exp(1))

Sigma <- rnorm(p ^ 2) %>%
    matrix(p, p) %>%
    (function(mat) t(mat) %*% mat)

round(Sigma, 3)
```

```
      [,1]   [,2]   [,3]
[1,] 3.360  1.209  2.472
[2,] 1.209  1.302 -0.518
[3,] 2.472 -0.518  4.497
```

```r
is_cov(Sigma)
```

```
[1] TRUE
```

Criando os dados, com mesma variância e vetores de média diferentes (também gerados aleatoriamente), teremos:

```r
muA <- rnorm(p, mean = -2)
muB <- rnorm(p, mean = 0)
nA <- 169
nB <- 196

linear <- rbind(
    MASS::mvrnorm(n = nA, mu = muA, Sigma = Sigma) %>%
      as_tibble() %>%
      cbind(Grupo = "A"),
    MASS::mvrnorm(n = nB, mu = muB, Sigma = Sigma) %>%
      as_tibble() %>%
      cbind(Grupo = "B")
)

pct80 <- as.integer(.8 * nrow(linear))

rows_train <- c(rep(TRUE, pct80), rep(FALSE, nrow(linear) - pct80)) %>%
  sample()

train_linear <- linear %>%
  filter(rows_train)

test_linear <- linear %>%
  filter(!rows_train)
```

```r
chart.Correlation(train_linear %>% dplyr::select(V1:V3))
```

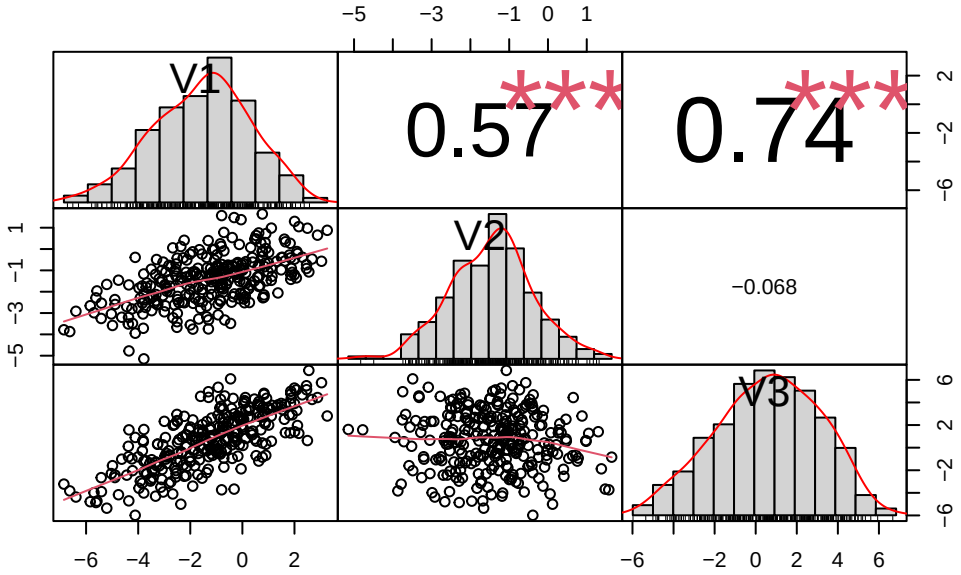


Figura 9: Correlações e histogramas das variáveis explicativas.

```r
with(train_linear, pairs(
  train_linear %>% dplyr::select(V1:V3),
  col = with(train_linear, c(A = "black", B = "pink")[Grupo]),
  upper.panel = function(...) {}
))
```

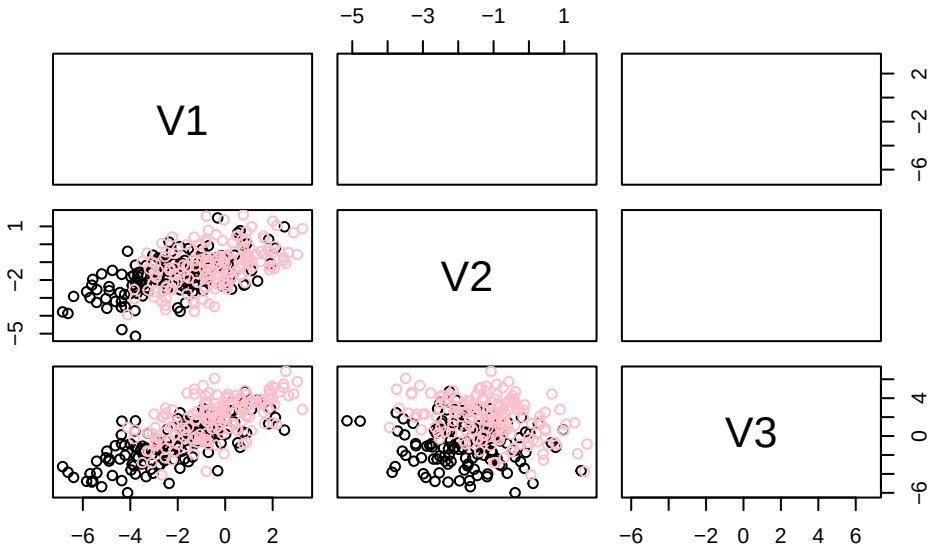


Figura 10: Gráficos de dispersão das variáveis explicativas com marcadores por classes linearmente separáveis.

Como esperado, muitos dados estão espalhados e em regiões sobrepostas, com várias delas correlacionadas, o que dá indícios da possibilidade de rotações.

A biblioteca `MASS` possui um comando pré-implementado que ajusta um LDA.

A principal forma de se definir o modelo é:

- `formula`: Sintaxe de fórmula análoga a qualquer outra modelagem em R.
- `data`: Dados a partir dos quais os coeficientes da fórmula serão ajustados. O argumento é opcional, caso cada elemento da fórmula esteja individualmente definido.

Existe também uma forma alternativa não muito usual:

- `x`: Conjunto de dados com variáveis explicativas;
- `grouping`: Variável resposta contendo as categorias a serem previstas.

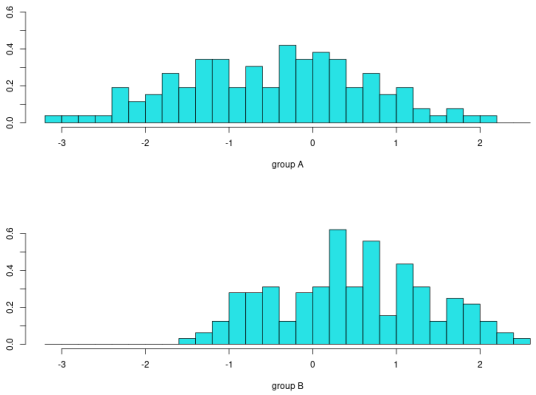
O método também permite selecionar as probabilidades de cada classe, métodos de estimação e se o modelo será treinado com validação cruzada ou não. Por padrão, ele não utiliza validação cruzada.

Ajustando três LDAs com apenas duas variáveis cada (e todos os argumentos default), obtemos:

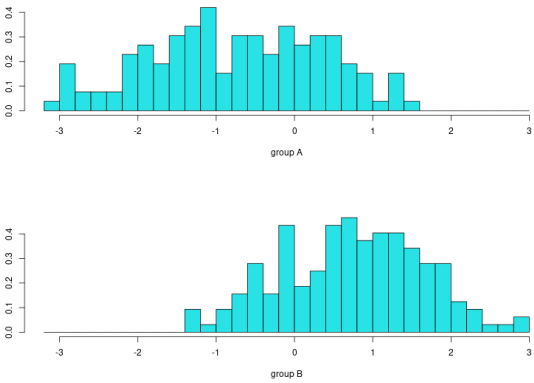
```r
incomplete_ldas <- list(
  V1V2 = lda(Grupo ~ V1 + V2, train_linear),
  V2V3 = lda(Grupo ~ V2 + V3, train_linear),
  V1V3 = lda(Grupo ~ V1 + V3, train_linear)
)

with(incomplete_ldas, V1V2)
```

```
Call:
lda(Grupo ~ V1 + V2, data = train_linear)

Prior probabilities of groups:
        A         B
0.4486301 0.5513699

Group means:
          V1        V2
A -2.1953259 -1.737232
B -0.4581957 -1.152510

Coefficients of linear discriminants:
          LD1
V1 0.54269666
V2 0.01850085
```

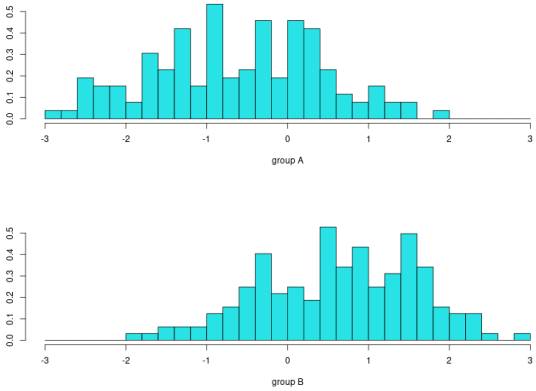
O resultado do método mostra a probabilidade de ocorrência de cada grupo (quando não definido manualmente, é utilizada a probabilidade nos dados), assim como a média e os coeficientes dos discriminantes. É possível plotar os resultados assim:



(a) Grupo ~ V1 + V2



(b) Grupo ~ V2 + V3



(c) Grupo ~ V1 + V3

Figura 11: Histogramas dos LDAs ajustados com duas variáveis explicativas.

Testando a acurácia com apenas duas variáveis explicativas, se obtém:

```
for (mdl in names(incomplete_ldas)) paste(
  "Acurácia:", mdl, accuracy(
    incomplete_ldas[[mdl]],
    test_linear,
    with(test_linear, Grupo)), '\n'
) %>%
  cat()
```

```
Acurácia: V1V2 0.589
Acurácia: V2V3 0.671
Acurácia: V1V3 0.644
```

```
(lda_full <- lda(Grupo ~ V1 + V2 + V3, train_linear))
```

```
Call:
lda(Grupo ~ V1 + V2 + V3, data = train_linear)

Prior probabilities of groups:
        A         B
0.4486301 0.5513699

Group means:
          V1        V2         V3
A -2.1953259 -1.737232 -0.7887385
B -0.4581957 -1.152510  1.8861528

Coefficients of linear discriminants:
         LD1
V1 -1.808266
V2  2.441663
V3  1.543961
```

```
paste("Acurácia:", accuracy(lda_full, test_linear, with(test_linear, Grupo))) %>%
  cat()
```

```
Acurácia: 0.904
```

Com apenas duas variáveis, nenhum dos classificadores atinge acurácia maior que 70%. Utilizando as três, o resultado ultrapassa 90% de acurácia.

Ajustando um modelo QDA para os mesmos dados (que não é o uso mais adequado da técnica, uma vez que as variâncias são iguais e eles são linearmente separáveis), o resultado é:

```
paste("Acurácia:", accuracy(
  qda(Grupo ~ V1 + V2 + V3, train_linear),
  test_linear, with(test_linear, Grupo))
) %>%
  cat()
```

```
Acurácia: 0.89
```

Que é um resultado inferior ao modelo LDA completo.

Para uma implementação do QDA, os grupos serão modificados a partir dos mesmos dados, forçando que a separação não-linear seja mais adequada que uma separação linear. Os resultados são expostos a seguir.

```
nonlinear <- linear %>%
  mutate(
    Grupo = ifelse(
      (
        abs(V1 ^ 2 * rgamma(n(), 3) * V3 - V2 ^ 3) >
```

```
          mean(-V1 + V2 - V3) / 2 + rnorm(n(), 10)
      ),
      "A", "B"
    )
  )

train_nonlinear <- nonlinear %>%
  filter(rows_train)

test_nonlinear <- nonlinear %>%
  filter(!rows_train)

with(train_nonlinear, pairs(
  train_nonlinear %>% dplyr::select(V1:V3),
  col = with(train_nonlinear, c(A = "black", B = "pink")[Grupo]),
  upper.panel = function(...) {}
))
```

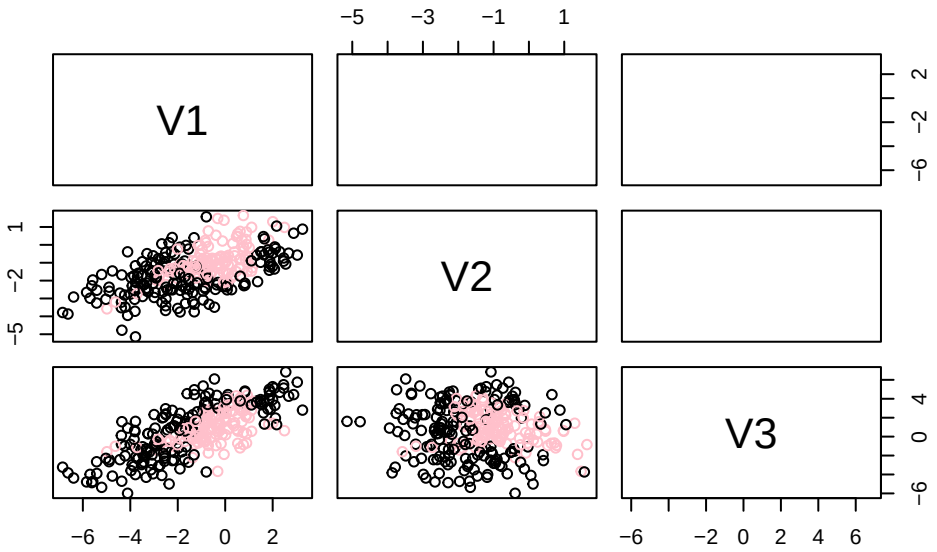


Figura 12: Gráficos de dispersão das variáveis explicativas com marcadores por classes não-linearmente separáveis.

Se observa, em todas as variáveis, regiões curvadas onde seria possível estabelecer limites de classificação. Ainda há algum nível de confusão entre as regiões, de forma que a separçaão não é perfeita.

Ajustando um QDA com as variáveis nessa estrutura, obtemos:

```
incomplete_qdas <- list(
  V1V2 = qda(Grupo ~ V1 + V2, train_linear),
```

```
  V2V3 = qda(Grupo ~ V2 + V3, train_linear),
  V1V3 = qda(Grupo ~ V1 + V3, train_linear)
)

for (mdl in names(incomplete_qdas)) paste(
  "Acurácia:", mdl, accuracy(
    incomplete_qdas[[mdl]],
    test_nonlinear,
    with(test_nonlinear, Grupo)), '\n'
) %>%
  cat()
```

```
Acurácia: V1V2 0.795
Acurácia: V2V3 0.767
Acurácia: V1V3 0.671
```

```
paste(
  "Acurácia:", accuracy(
    qda(Grupo ~ V1 + V2 + V3, train_nonlinear),
    test_nonlinear,
    with(test_nonlinear, Grupo)), '\n'
) %>%
  cat()
```

```
Acurácia: 0.932
```

Para estes dados, uma das classificações com duas variáveis (V1 com V2) já chegou a uma acurácia próxima de 80%. Utilizando as três, mais de 93% dos pontos são classificados corretamente.

Ajustando um LDA para os dados não-lineares, a título de comparação, se obtém:

```
accuracy(
  lda(Grupo ~ V1 + V2 + V3, train_nonlinear),
  test_nonlinear, with(test_nonlinear, Grupo)
)
```

```
[1] 0.767
```

Que é uma precisão mais baixa que a precisão do QDA, como esperado pelo fato de a aplicação LDA não ser o uso mais adequado do método para esses dados.

Apesar disso, mesmo não sendo o uso mais adequado, o ajuste LDA ainda acertou mais de 75% das classificações.