Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

28 abril 2024

Entrega 2

Prof. Dr. George von Borries

Aluno: Bruno Gondim Toledo

Matrícula: 15/0167636

Tópicos 2

1º/2024

3)

a)

```
mu1 <- c(1, 0)
mu2 <- c(-1, 0)

sigma <- matrix(c(1, 0,
                  0, 1),2)

#set.seed(150167636)
df = data.frame(MASS::mvrnorm(100, mu1, sigma))
#set.seed(150167636)
df = rbind(df,data.frame(MASS::mvrnorm(100, mu2, sigma)))
df$grupo = factor(c(rep(1,100),rep(2,100)))
head(df)
```

```
##           X1         X2 grupo
## 1  2.4552392  0.6722484     1
## 2  1.6777102 -1.5438905     1
## 3 -0.3412950  0.5780403     1
## 4  1.5394537  0.4350617     1
## 5 -0.4627649 -0.1107235     1
## 6  2.3508733 -0.3817968     1
```

```
tail(df)
```

```
##             X1         X2 grupo
## 195 -0.7865031 -1.1502757     2
## 196 -1.9438833  0.6473254     2
## 197 -1.7957555 -0.6307067     2
## 198 -1.3701111  0.9196888     2
## 199 -1.2954558  1.2282873     2
## 200 -1.7159328 -0.4232991     2
```

b)

```r
shapiro.test(df$X1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$X1
## W = 0.98977, p-value = 0.1652
```

```r
shapiro.test(df$X2)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$X2
## W = 0.99123, p-value = 0.268
```

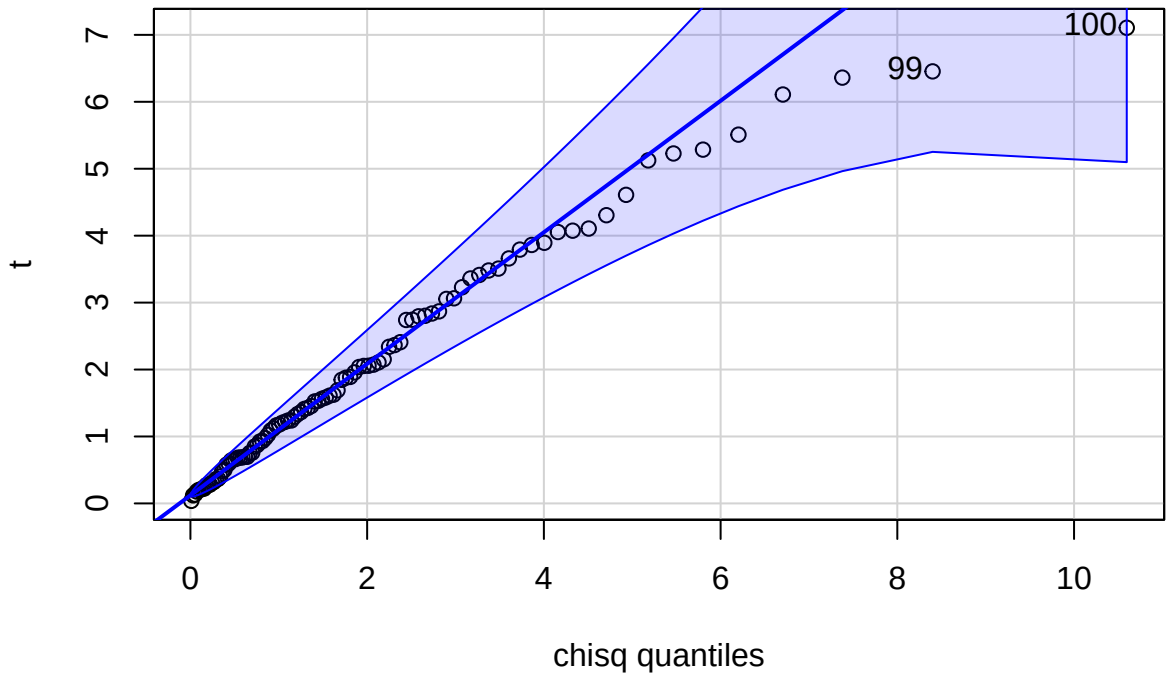
```r
x1 = df |> filter(grupo == 1) |> dplyr::select(X1) |> pull()
shapiro.test(x1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x1
## W = 0.98763, p-value = 0.4811
```

```r
x2 = df |> filter(grupo == 1) |> dplyr::select(X2) |> pull()
shapiro.test(x2)
```

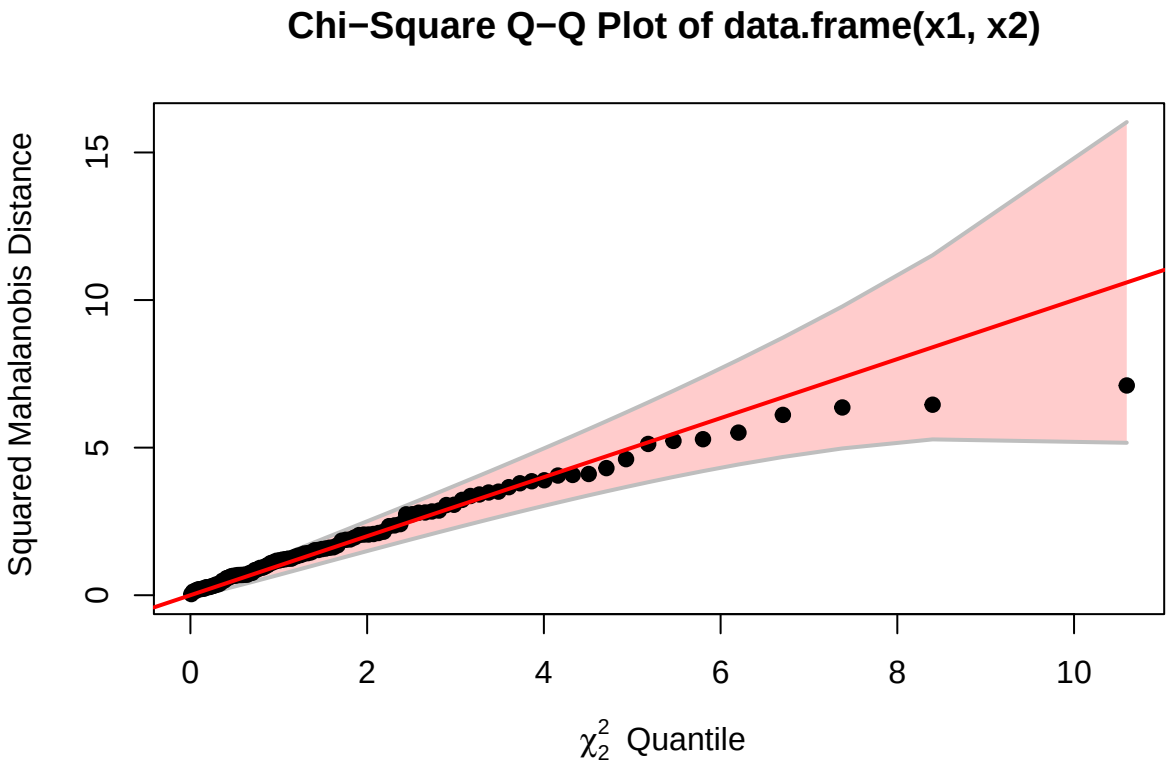
```
##
##  Shapiro-Wilk normality test
##
## data:  x2
## W = 0.99342, p-value = 0.9122
```

```r
mu <- t(matrix(c(mean(x1),mean(x2)),1,2))
S <- matrix(c(var(x1),cov(x1,x2),
              cov(x1,x2),var(x2)),2,2)
Sinv <- solve(S)

distancias <- vector("numeric", length(x1))
for (i in 1:length(x1)) {
  xjx <- c(x1[i], x2[i]) - mu
  distancia <- t(xjx) %*% Sinv %*% xjx
  distancias[i] <- distancia
}
#distancias

limite <- qchisq(.5, df = 2)
prop1 <- sum(distancias < limite)/ length(distancias)

t <- sort(distancias)
car::qqPlot(t, dist="chisq", df=2)
```




```
## [1] 100  99
```

```
heplots::cqplot(data.frame(x1,x2))
```



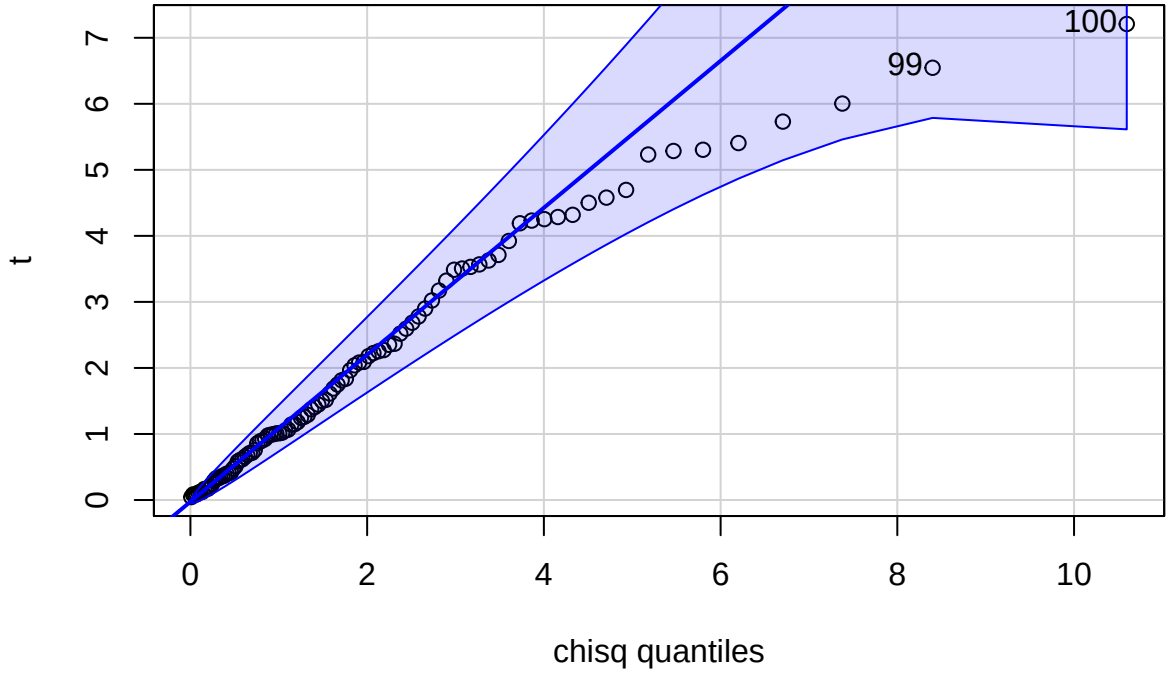

```
x1 = df |> filter(grupo == 2) |> dplyr::select(X1) |> pull()
shapiro.test(x1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x1
## W = 0.99039, p-value = 0.6962
```

```r
x2 = df |> filter(grupo == 2) |> dplyr::select(X2) |> pull()
shapiro.test(x2)
```

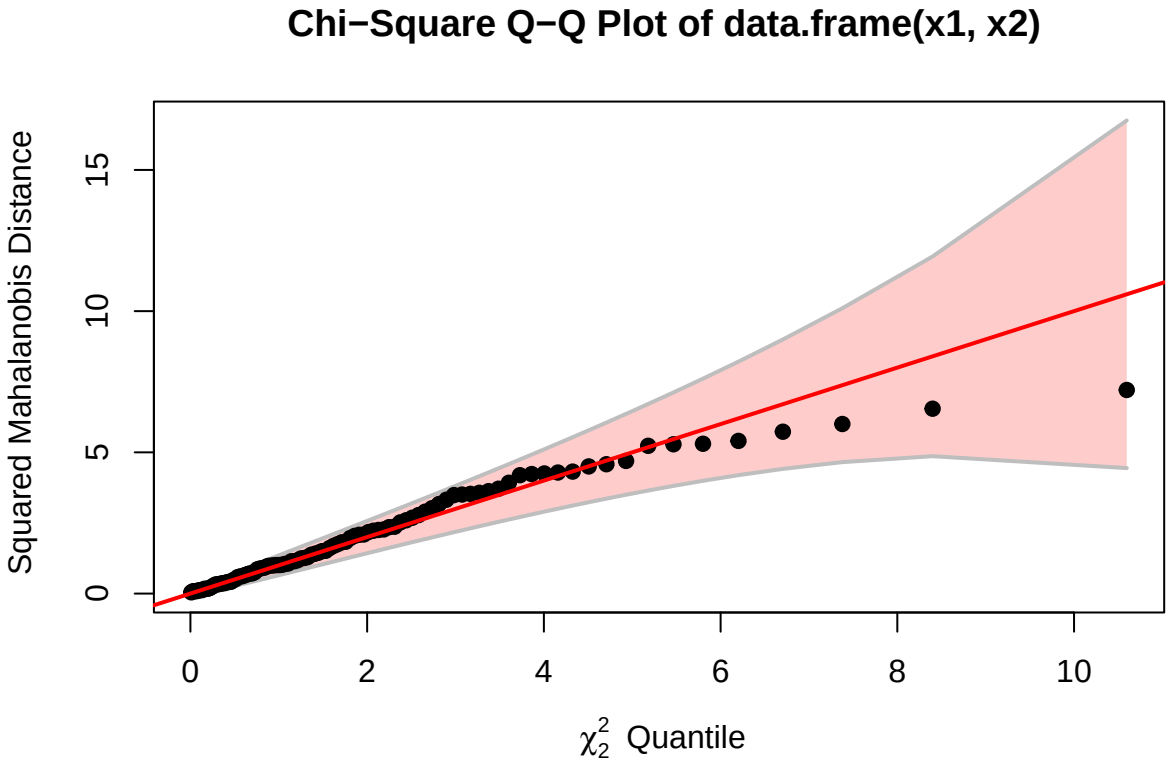
```
##
##  Shapiro-Wilk normality test
##
## data:  x2
## W = 0.97391, p-value = 0.04434
```

```r
mu <- t(matrix(c(mean(x1),mean(x2)),1,2))
S <- matrix(c(var(x1),cov(x1,x2),
              cov(x1,x2),var(x2)),2,2)
Sinv <- solve(S)

distancias <- vector("numeric", length(x1))
for (i in 1:length(x1)) {
  xjx <- c(x1[i], x2[i]) - mu
  distancia <- t(xjx) %*% Sinv %*% xjx
  distancias[i] <- distancia
}
#distancias

prop2 <- sum(distancias < limite)/ length(distancias)

t <- sort(distancias)
car::qqPlot(t, dist="chisq", df=2)
```




```
## [1] 100  99
```

```r
heplots::cqplot(data.frame(x1,x2))
```

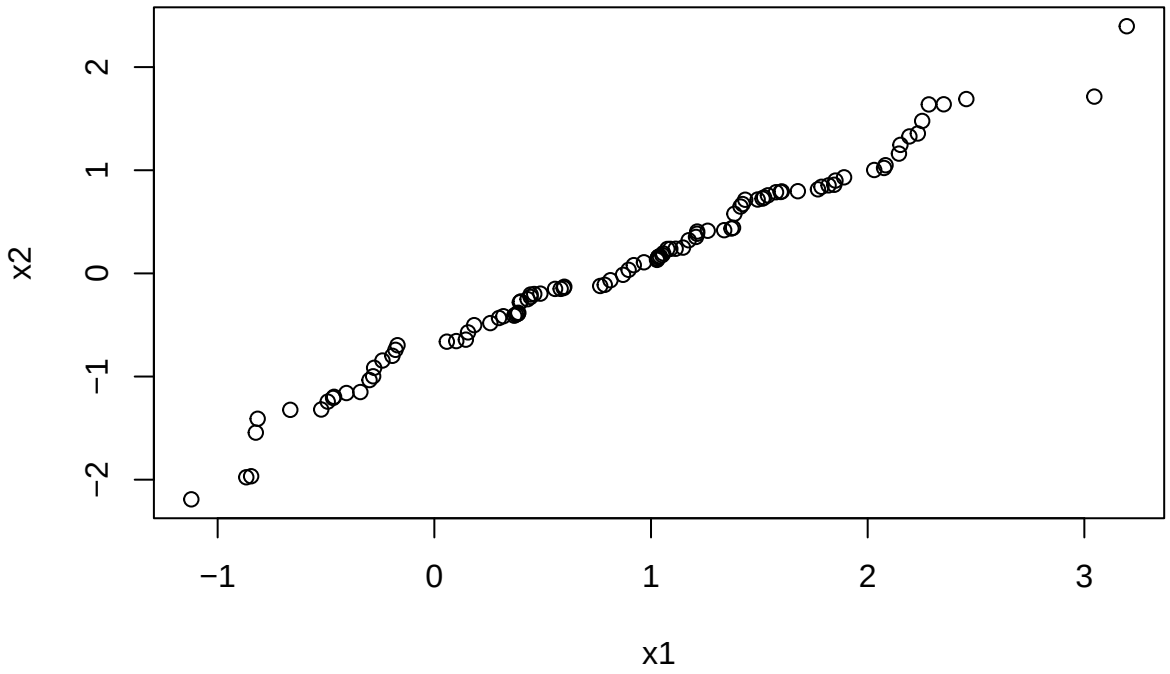


Os testes de shapiro-wilk não rejeitaram a normalidade de nenhuma variável à $\alpha = 0,05$. Além disso, observando os quantis da distribuição χ^2_2 não é possível rejeitar a normalidade multivariada dos dados.

```
x1 = df |> filter(grupo == 1) |> dplyr::select(X1) |> pull()
x2 = df |> filter(grupo == 1) |> dplyr::select(X2) |> pull()
knitr::kable(MVN::mvn(data.frame(x1,x2))$multivariateNormality)
```

Test	HZ	p value	MVN
Henze-Zirkler	0.5365286	0.5174029	YES

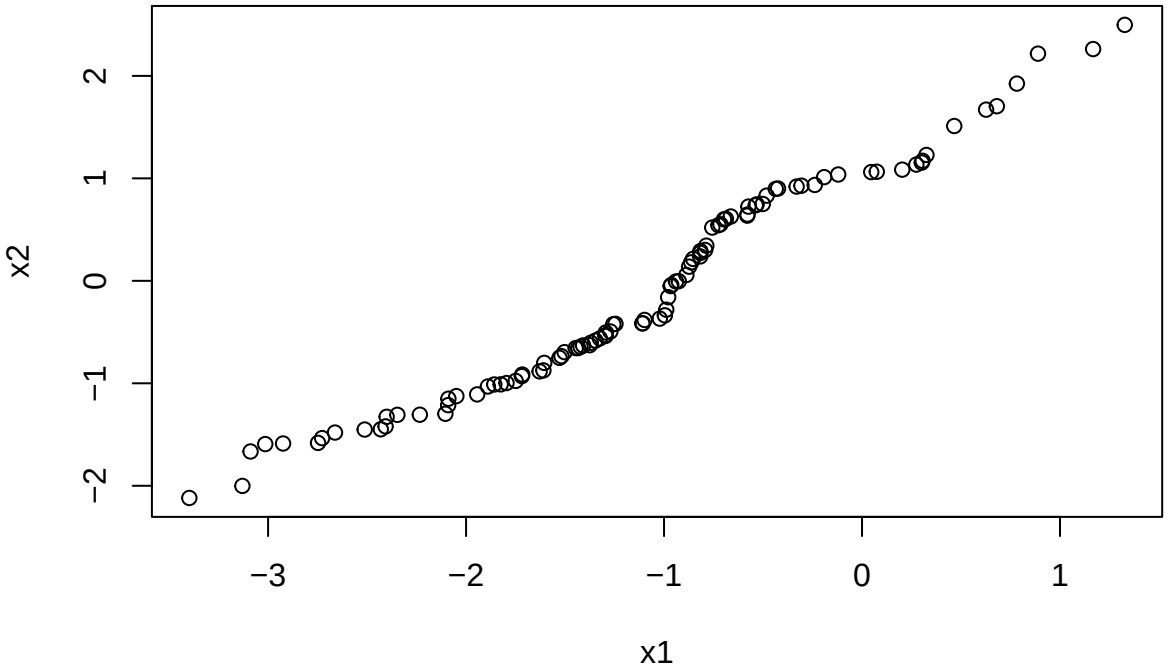
```
qqplot(x=x1,y=x2)
```



```r
x1 = df |> filter(grupo == 2) |> dplyr::select(X1) |> pull()
x2 = df |> filter(grupo == 2) |> dplyr::select(X2) |> pull()
knitr::kable(MVN::mvn(data.frame(x1,x2))$multivariateNormality)
```

Test	HZ	p value	MVN
Henze-Zirkler	0.6468004	0.3107423	YES

```r
qqplot(x=x1,y=x2)
```



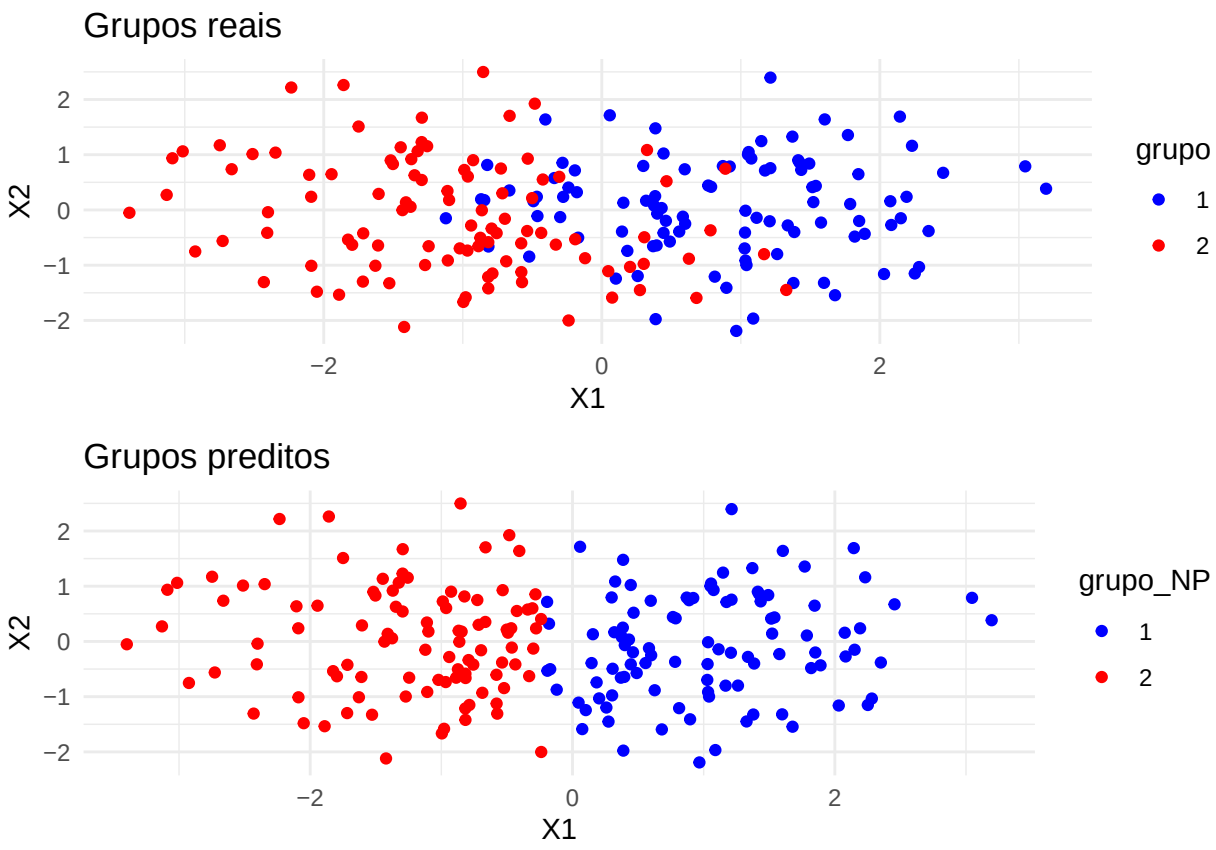
Testes extras contidos no pacote **MVN** também reforçam a hipótese nula de normalidade multivariada dos dados. O gráfico qq também reforça a hipótese.

c)

```
v = numeric()
p = 2
S = sigma
for (i in 1:200){
  x = c(df$X1[i], df$X2[i])
  v = append(v,(1/(2*pi)^(p/2)*det(S)^(1/20)*exp((-t((x-mu1)) %*% solve(S) %*% (x-mu1))/2)) / (1/(2*
}
threshold = quantile(v, probs = 0.5)
df$v = v
df$grupo_NP <- factor(ifelse(df$v > threshold, 1, 2))

df |> mutate(acerto = grupo == grupo_NP) |> summarise(acertos = sum(acerto), porcentagem = acertos/2
```

```
##   acertos porcentagem
## 1     168        0.84
```

```
plot1 <- ggplot(df, aes(x = X1, y = X2, color = grupo)) +
  geom_point() +
  scale_color_manual(values = c("blue", "red")) +
  labs(title = "Grupos reais", x = "X1", y = "X2") +
  theme_minimal()
plot2 <- ggplot(df, aes(x = X1, y = X2, color = grupo_NP)) +
  geom_point() +
  scale_color_manual(values = c("blue", "red")) +
  labs(title = "Grupos preditos", x = "X1", y = "X2") +
  theme_minimal()

gridExtra::grid.arrange(plot1, plot2, nrow = 2)
```



Daqui, vemos que as regiões Ω_1 e Ω_2 foram definidas em função apenas de X1. É um resultado esperado, se

lembrarmos que os vetores de média $\mu_{\omega_1} = [1, 0]^T$ e $\mu_{\omega_2} = [-1, 0]^T$, com matriz de variância-covariâncias iguais. Portanto, a "fronteira" foi colocada no quantil 0,5 da razão de verossimilhanças, que no caso se aproxima de $X1 = 0$. Ou seja, para $X1 \in (-\infty, 0)$ o grupo predito é 2 e para $X1 \in (0, \infty)$ o grupo predito é 1,$\forall X2 \in \mathbb{R}$.

d)

```
for(i in 1:10){
  x = rnorm(2)
  print(x)
  v = (1/(2*pi)^(p/2)*det(S)^(1/20)*exp((-t((x-mu1)) %*% solve(S) %*% (x-mu1))/2)) / (1/(2*pi)^(p/2)
  if (v > threshold) {
    print("A coordenada x pertence a Omega_1")
  } else {
    print("A coordenada x pertence a Omega_2")
  }
}
```

```
## [1] -1.371770 -1.115516
## [1] "A coordenada x pertence a Omega_2"
## [1] -0.08483967  0.19849119
## [1] "A coordenada x pertence a Omega_1"
## [1] -0.5545097  0.8141334
## [1] "A coordenada x pertence a Omega_2"
## [1]  0.6064418 -0.8869894
## [1] "A coordenada x pertence a Omega_1"
## [1] -0.4156459  1.9660901
## [1] "A coordenada x pertence a Omega_2"
## [1] 0.1990475 0.5999252
## [1] "A coordenada x pertence a Omega_1"
## [1] -0.499790  0.564056
## [1] "A coordenada x pertence a Omega_2"
## [1] -0.1074565  0.6241914
## [1] "A coordenada x pertence a Omega_1"
## [1] -0.1820585 -1.4993047
## [1] "A coordenada x pertence a Omega_1"
## [1]  0.4598199 -0.6605023
## [1] "A coordenada x pertence a Omega_1"
```

Desta, reforça-se o inferido em c). A fronteira de decisão se aproxima de $X1 = 0$.

4)

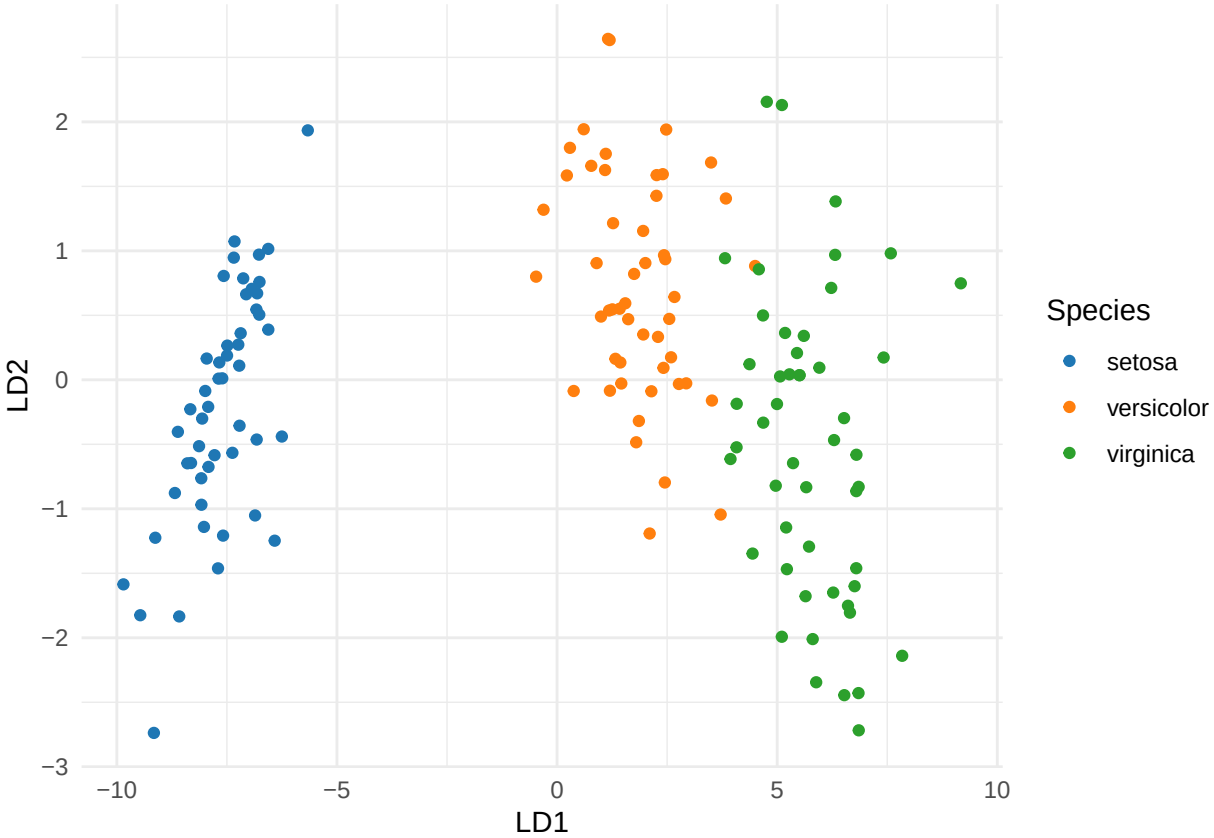
5)

Utilizando do conjunto de dados *iris*, irei realizar uma análise de discriminante em *R* e outra em *Julia*.

Não irei separar o conjunto em treino-teste. Farei a análise de discriminante linear (LDA) diretamente no conjunto de dados, apenas para comparar a implementação em *R* e *Julia*.

Em *R*, poderíamos realizar a análise de discriminantes com o seguinte código, utilizando dos pacotes *caret* e *MASS*: [1]

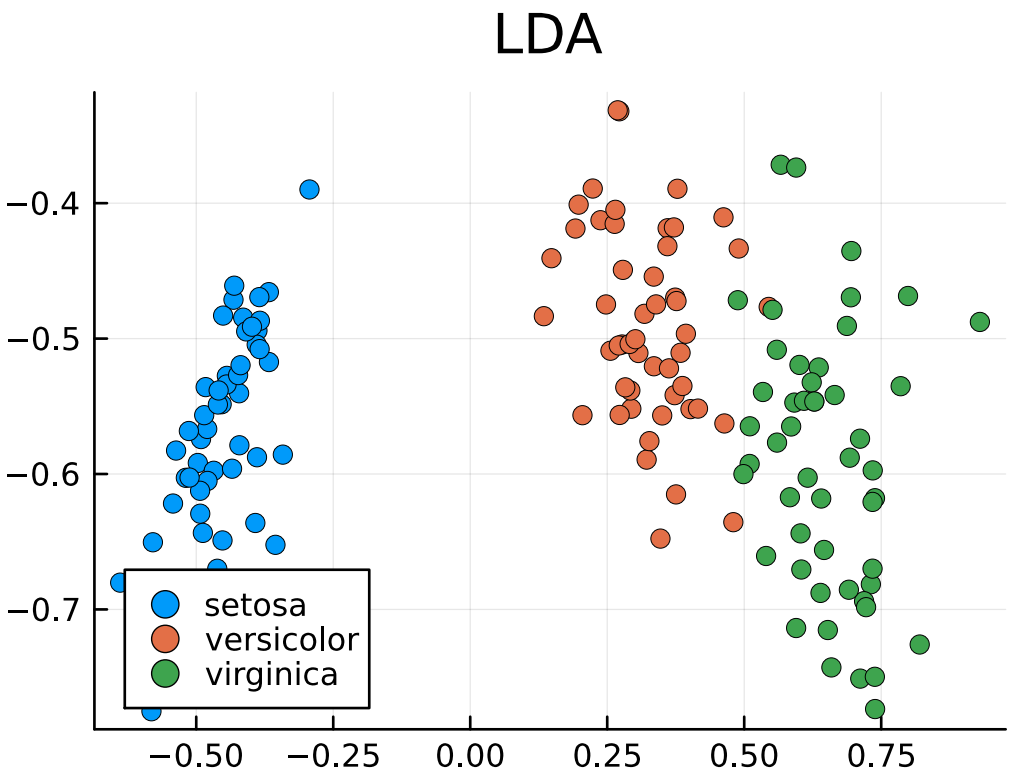
```
preproc.param <- iris %>%
  preProcess(method = c("center", "scale"))

iris.transformed <- preproc.param %>%
  predict(iris)

model <- lda(Species~., data = iris.transformed)

lda.data <- cbind(iris.transformed, predict(model)$x)
lda.data$LD1 <- lda.data$LD1 * -1

ggplot(lda.data, aes(LD1, LD2)) +
  geom_point(aes(color = Species)) +
  scale_color_manual(values = c("setosa" = "#1F77B4",
                                "versicolor" = "#FF7F0E",
                                "virginica" = "#2CA02C")) +
  theme_minimal()
```



Com *Julia*, poderiamos realizar uma análise análoga com o seguinte código: [2]

```
using MultivariateStats, RDatasets, Plots

iris = dataset("datasets", "iris")

X = Matrix(iris[1:end,1:4])'
X_labels = Vector(iris[1:end,5])

lda = fit(MulticlassLDA, X, X_labels; outdim=2)
Ylda = predict(lda, X)

p = plot(size=(400,300))

for s in ["setosa", "versicolor", "virginica"]
    points = Ylda[:,X_labels.==s]
    scatter!(p, points[1,:],points[2,:], label=s, legend=:bottomleft)
end
```

```
plot!(p, title="LDA")
```



Referências:

[1] Discriminant Analysis Essentials in R. http://www.sthda.com/english/articles/36-classification-methods-essentials/146-discriminant-analysis-essentials-in-r/ Acessado em: 28/04/2024, 14:13.

[2] Linear Discriminant Analysis. https://juliastats.org/MultivariateStats.jl/dev/lda/ Acessado em: 28/04/2024, 14:13.