Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

30 abril 2024

Entrega 2

Prof. Dr. George von Borries

Aluno: Bruno Gondim Toledo

Matrícula: 15/0167636

Aluno: Stefan Zurman Gonçalves

Matrícula: 19/0116994

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  2.41285244  0.01728087     1
## 2  1.07586326 -0.18514876     1
## 3 -0.03338347  0.52265767     1
## 4  1.04772660  0.27411671     1
## 5  0.85094091 -1.33702516     1
## 6  1.01295504  0.42339548     1
```

```
tail(df)
```

```
##              X1         X2 grupo
## 195 -1.08204138  1.0224481     2
## 196 -0.28704589 -1.3562125     2
## 197  0.02948022  0.4892296     2
## 198 -1.57711938  0.5615401     2
## 199 -0.88565877  1.7557833     2
## 200  1.31813053  1.6547504     2
```

b)

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

```
##
##  Shapiro-Wilk normality test
##
## data:  df$X1
## W = 0.98464, p-value = 0.02817
```

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

```
##
##  Shapiro-Wilk normality test
##
## data:  df$X2
## W = 0.9885, p-value = 0.107
```

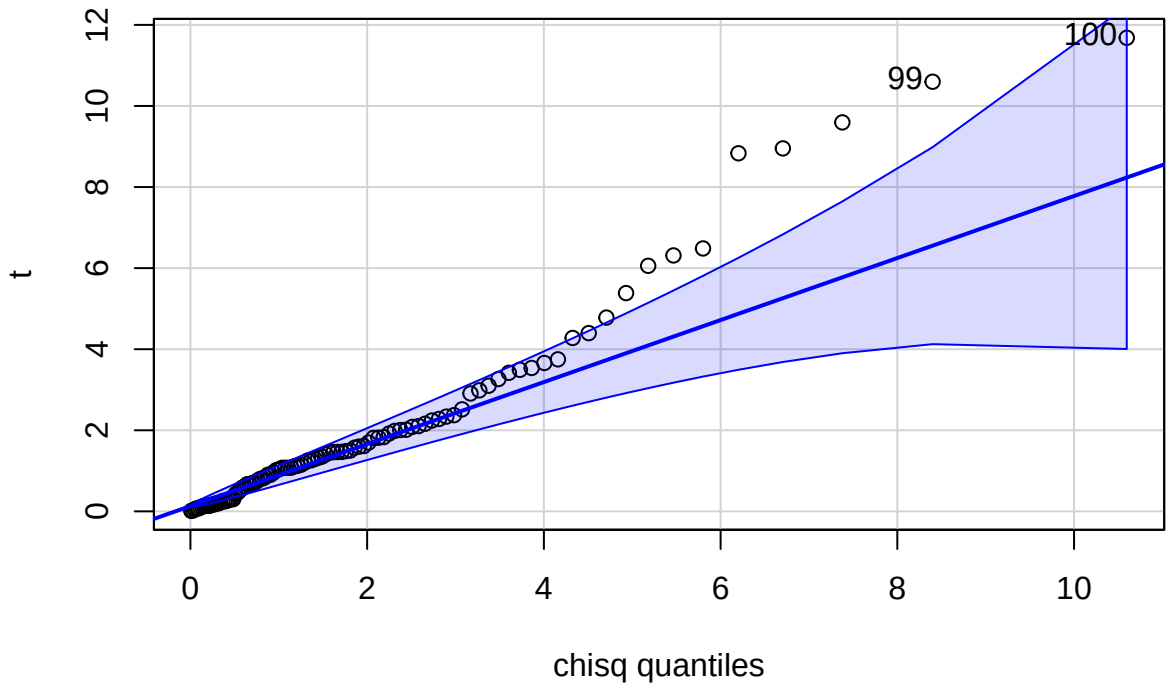
```r
x1 = df |> filter(grupo == 1) |> dplyr::select(X1) |> pull()
shapiro.test(x1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x1
## W = 0.98817, p-value = 0.5207
```

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

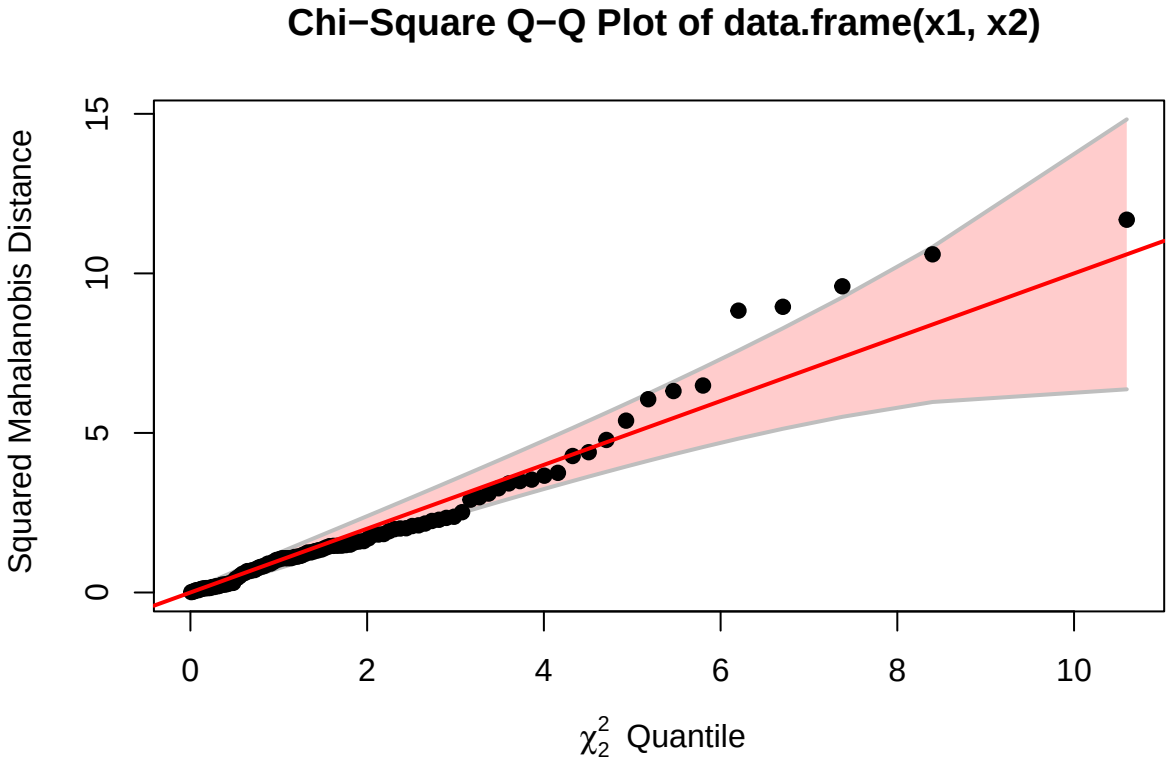
```
##
##  Shapiro-Wilk normality test
##
## data:  x2
## W = 0.98081, p-value = 0.1537
```

```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)
```

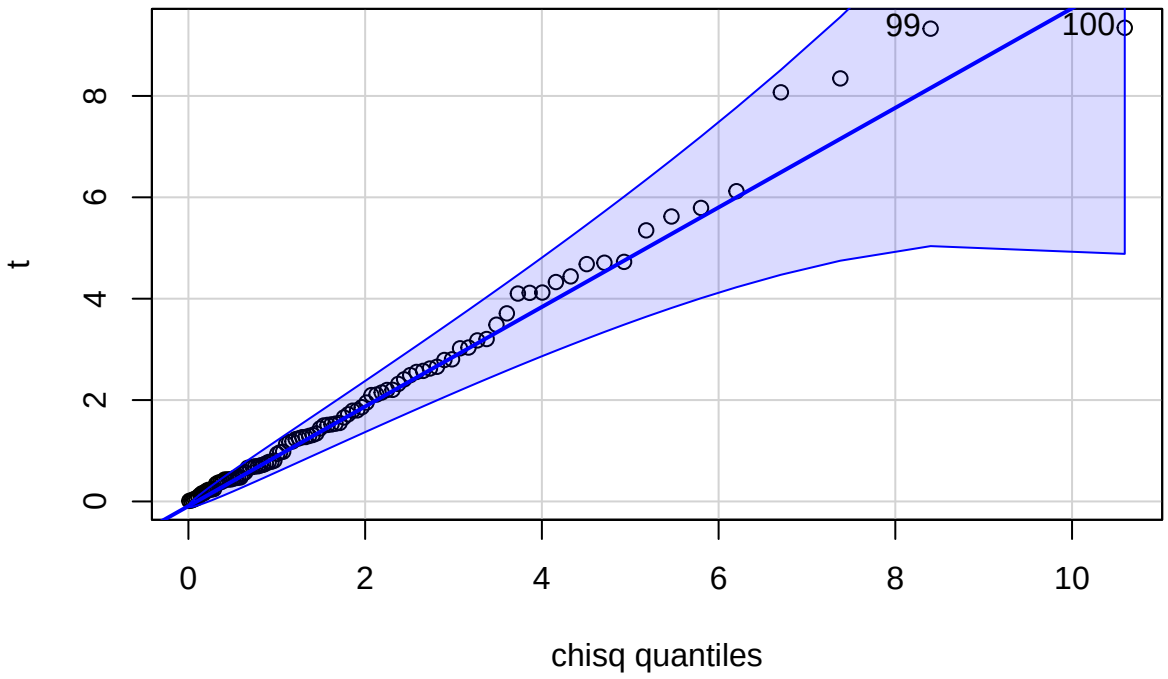
```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  x1
## W = 0.98593, p-value = 0.3695
```

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

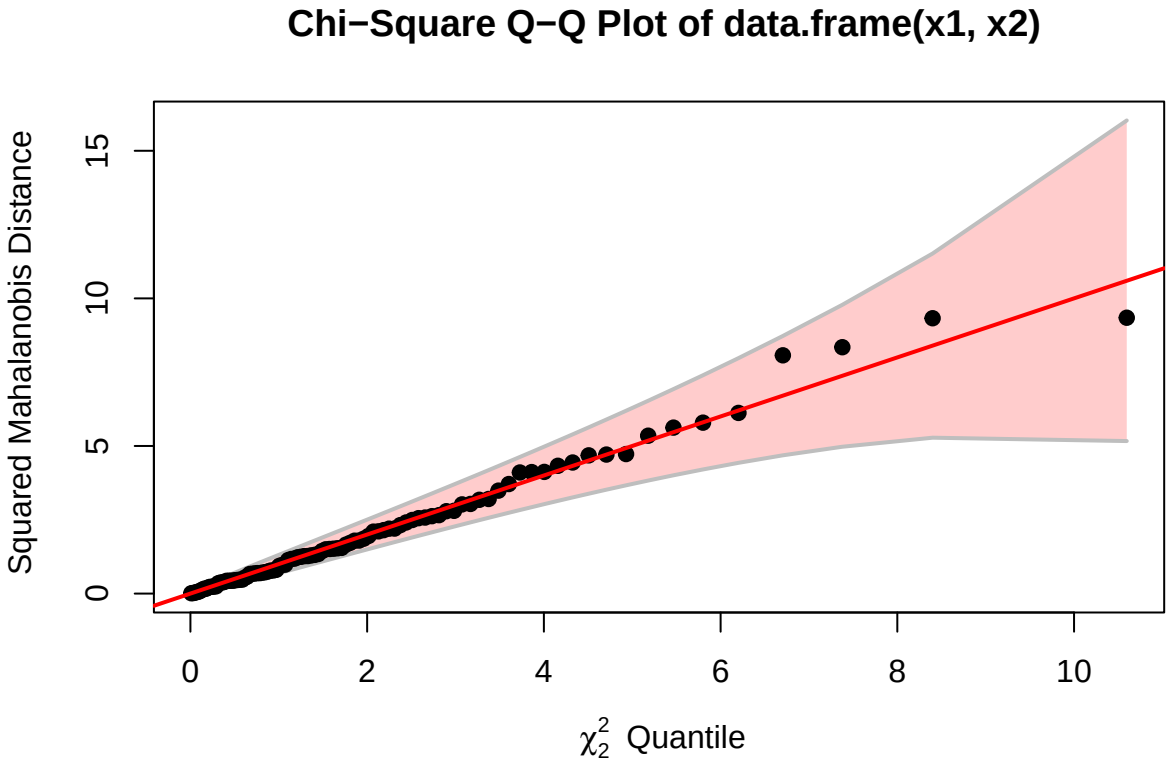
```
##
##  Shapiro-Wilk normality test
##
## data:  x2
## W = 0.98952, p-value = 0.6256
```

```
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
```

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

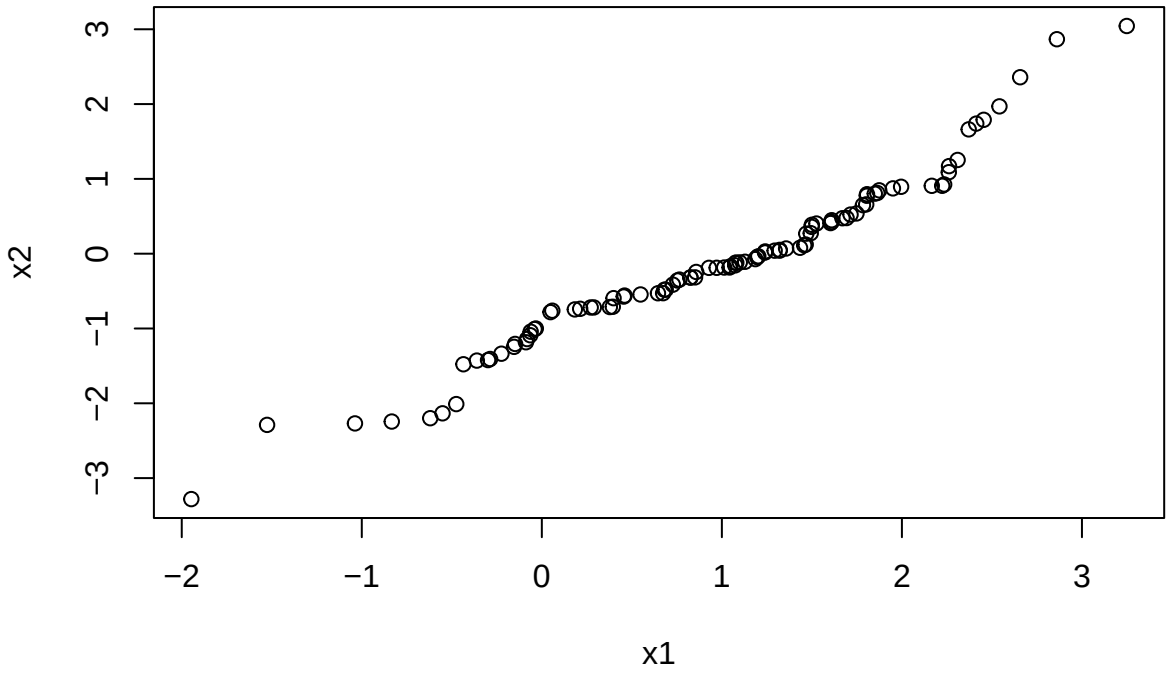


Os testes de shapiro-wilk não rejeitaram a normalidade de nenhuma variável à $\alpha = 0,05$. Além disso, observando os quantis da distribuição χ^2_2 não é possível rejeitar a normalidade multivariada dos dados.

```
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	0.623553	0.3488188	YES

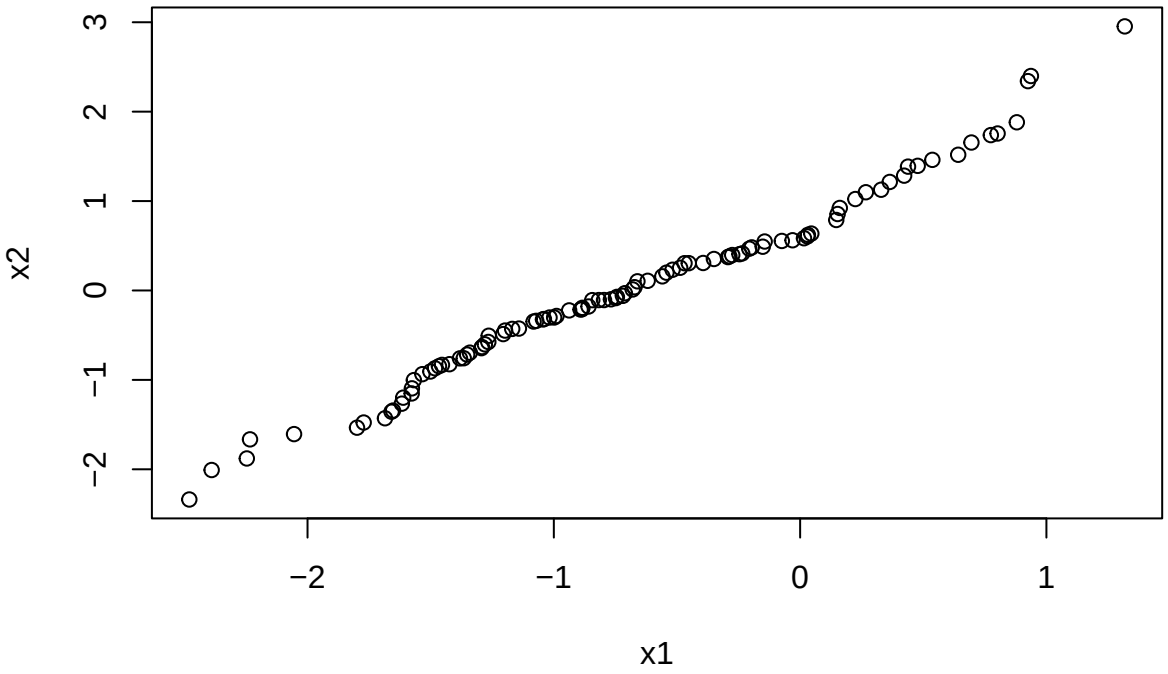
```
qqplot(x=x1,y=x2)
```



```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	0.4882708	0.6234679	YES

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



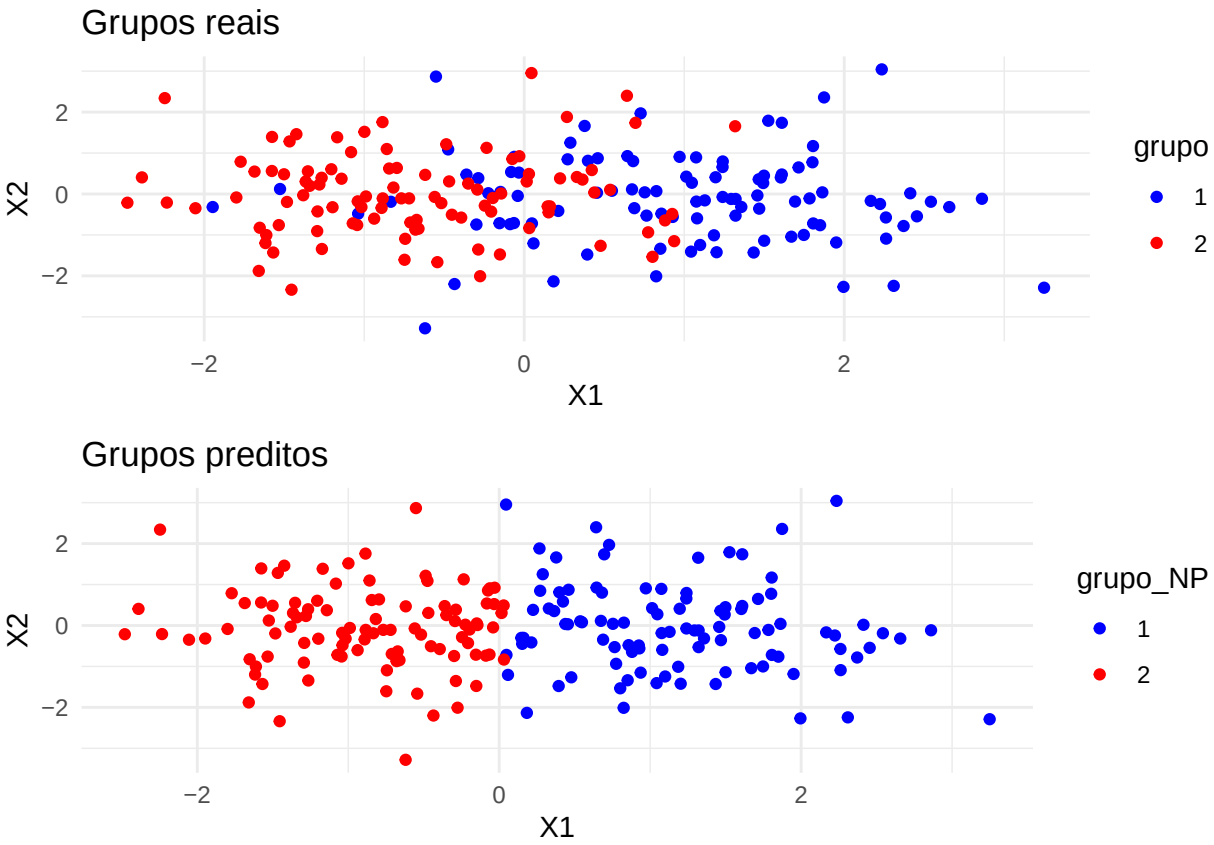
Testes extras contidos no pacote **MVN** também reforçam a hipótese nula de normalidade multivariada dos dados. O gráfico qq também reforça a hipótese.

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)^(1/2)*exp((-t((x-mu1)) %*% solve(S) %*% (x-mu1))/2)) /
               (1/(2*pi)^(p/2)*det(S)^(1/2)*exp((-t((x-mu2)) %*% solve(S) %*% (x-mu2))/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/200)
```

```
##   acertos porcentagem
## 1     160         0.8
```

```
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)^(1/2)*exp((-t((x-mu1)) %*% solve(S) %*% (x-mu1))/2)) /
    (1/(2*pi)^(p/2)*det(S)^(1/20)*exp((-t((x-mu2)) %*% solve(S) %*% (x-mu2))/2))
  if (v > threshold) {
    print("A coordenada x pertence a Omega_1")
  } else {
    print("A coordenada x pertence a Omega_2")
  }
}
```

```
## [1] -0.9042236 -0.2708688
## [1] "A coordenada x pertence a Omega_2"
## [1]  0.5710214 -0.2859604
## [1] "A coordenada x pertence a Omega_1"
## [1] -0.03322718 -0.35287406
## [1] "A coordenada x pertence a Omega_2"
## [1]  1.2119692 -0.8970619
## [1] "A coordenada x pertence a Omega_1"
## [1] 0.1078381 0.7982486
```

```
## [1] "A coordenada x pertence a Omega_1"
## [1] -0.8332342 -0.8641457
## [1] "A coordenada x pertence a Omega_2"
## [1] -0.8875222 -1.4347958
## [1] "A coordenada x pertence a Omega_2"
## [1]  0.01862025 -0.53906169
## [1] "A coordenada x pertence a Omega_2"
## [1] -1.43178  1.42090
## [1] "A coordenada x pertence a Omega_2"
## [1]  2.198684 -1.129944
## [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)

$$\mathbb{P}(\mathbf{x}|\omega_1) \sim \mathbf{N_p}(\mu_1, \Sigma)$$

$$\mathbb{P}(\mathbf{x}|\omega_2) \sim \mathbf{N_p}(\mu_2, \Sigma)$$

Temos

$$f_1(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)$$

e

$$f_2(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2)$$

para um vetor de características $\mathbf{x}$.

Assim:

$$\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = \frac{exp\{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)\}}{exp\{-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2)\}}$$

Ainda,

$$\mathcal{L}(\mathbf{x}) = ln(\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}) = -\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1) + \frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2)$$

$$= -\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_1^T\Sigma^{-1}(\mathbf{x}-\mu_1) + \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_2^T\Sigma^{-1}(\mathbf{x}-\mu_2)$$

$$= \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mu_2 + \frac{1}{2}\mu_1^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_2^T\Sigma^{-1}\mathbf{x} + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2$$

Como $(\mathbf{x}^T\Sigma^{-1}\mu_1)^T$ e $(\mathbf{x}^T\Sigma^{-1}\mu_2)^T$ são escalares, $\mathbf{x}^T\Sigma^{-1}\mu_1 = (\mathbf{x}^T\Sigma^{-1}\mu_1)^T$, e $\mathbf{x}^T\Sigma^{-1}\mu_2 = (\mathbf{x}^T\Sigma^{-1}\mu_2)^T = \mu_2^T\Sigma^{-1}\mathbf{x}$, uma vez que também Σ^{-1} é simétrico. Assim,

$$\mathcal{L}(\mathbf{x}) = \frac{1}{2}\mu_1^T\Sigma^{-1}\mathbf{x} + \frac{1}{2}\mu_1^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_2^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_2^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2$$

$$= -\frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1 - \mu_2^T\Sigma^{-1}\mu_2) + (\mu_1 - \mu_2)^T\Sigma^{-1}\mathbf{x}$$

$= \beta_0 + \beta^T\mathbf{x}$. Logo, $\mathcal{L}(\mathbf{x})$ é linear em relação ao vetor de características $\mathbf{x}$ □

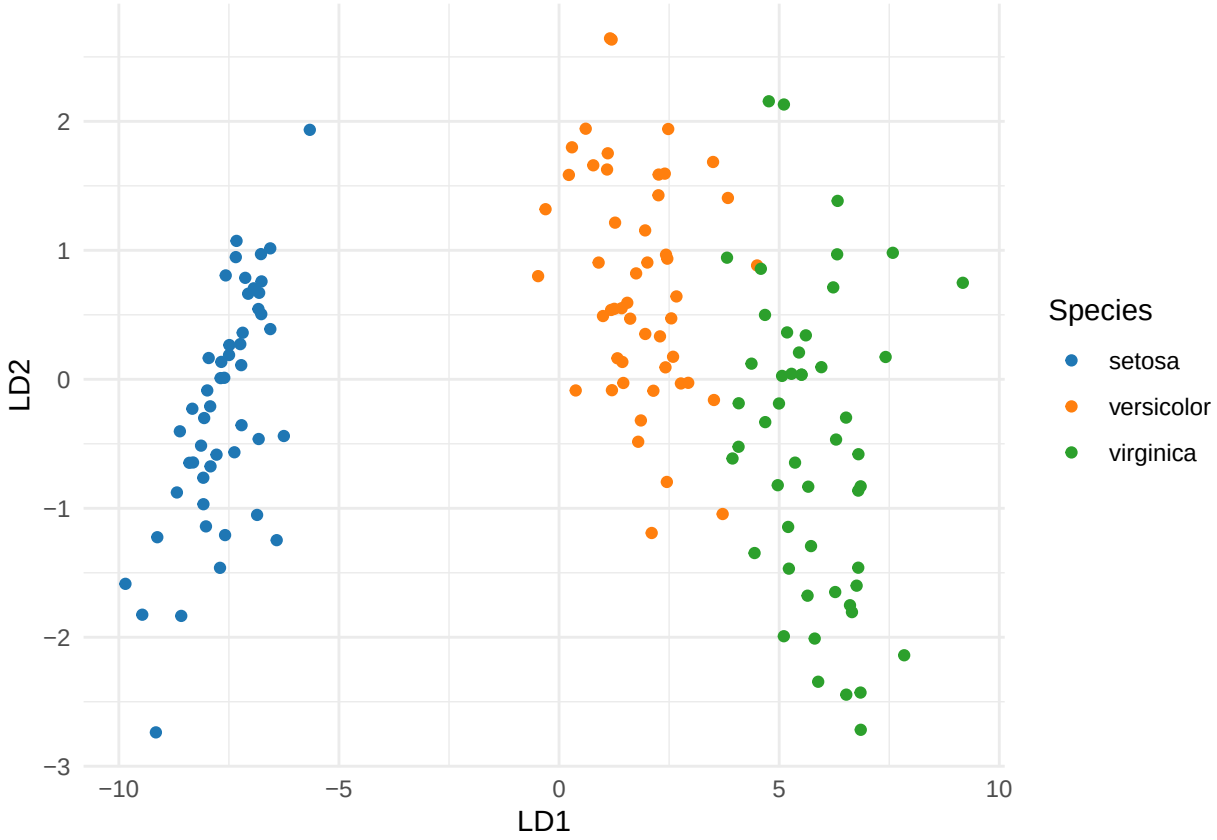
5)

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]

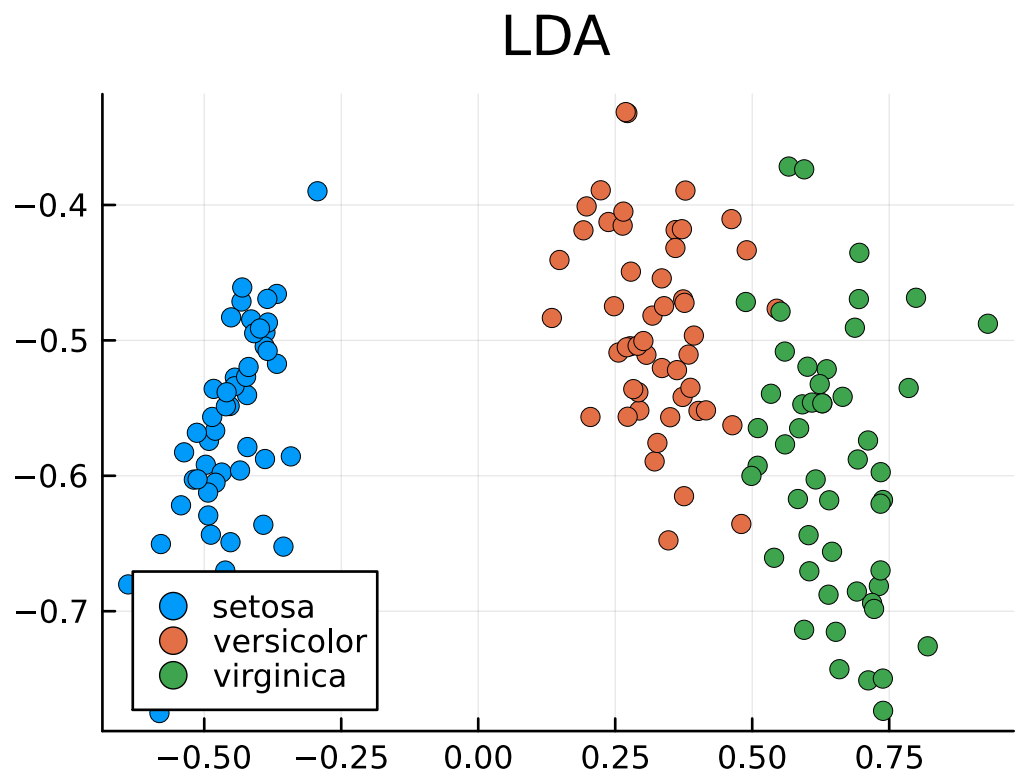
```
preproc.param <- iris %>%
  preProcess(method = c("center", "scale"))

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.